COULD "LOCK OUT" BASIC SO NOTHING COULD HAPPEN, THAT IS
WHY WE MUST CHECK FOR ^C HERE, IF THIS IS THE CASE
THEN WE MUST TAKE THE SAME ACTION AS FOR THE CASE WHERE
WE COULD PHYSICALLY DETECT THE FACT THAT THE CLOCK WAS
RUNNING TOO FAST. WHEN WE DECIDE THAT THE CLOCK IS RUNNING TOO FAST,
WE STOP THE CLOCK (TO PREVENT FURTHER CLOCK INTERUPTS) AND
CALL 'RTERR' TO PRINT OUT THE ERROR MESSAGE ["RATE ERROR"];
IF THERE WAS NO ERROR, THEN WE EXIT NORMALLY VIA 'INTEXT'.

APUT: THIS PUTS A SAMPLE IN THE BUFFER, IF NO ROOM, WE GET
A BUFFER FULL MESSAGE, POINTERS ARE SET UP BY REAL TIME STATEMENT.

AGET: OPPOSITE OF APUT, IF NOTHING THERE, IT PUTS BASIC TO
SLEEP UNTIL AN INTERUPT OCCURS', DESCRIBED PREVIOUSLY,

ABOP: THIS BOPS UP POINTERS OF APUT AND AGET, LIMITS SET BY
REAL TIME STATEMENT, ADA1,ADA2 AND ADA3 ARE DETERMINING
LIMITS AND FACTORS,

UCLS: CLS FUNCTION, PICKS UP CLKSTS AND RETURNS IT,

UCLC: EXECUTES 6137 AND RETURNS RESULTS IN FAC,

UUULLL: THIS IS THE UPPER CORE RESETER',

DOAD: TAKES AC AS CHANNEL NUMBER AND DOES CONVERSION, HAS
TIME OUT INCASE NO A-D OR A-D NOT WORKING,

USETF: THIS ROUTINE TAKES THE AC AND MAKES IT INTO A GOOD FAC,
IT CALLS BEGFIX AND ANORM,

UUAC1-3: THESE ROUTINES ADD AC1 -AC3
RESPECTIVELY TO THE AC, THEY GET IT FROM FIELD Ø,

UUMEVAL: MEVAL FUDGE CALL

UUFIX: FIX FUDGE CALL,

UUDEVC: DEVCON FUDGE CALL,

UUJMS: UPPER CORE COUNTER PART OF LLLJMS

UUJMP: UPPER CORE COUNTER PART OF LLLJMP

UPCOMDO: UPPER CORE COMMAND DISPATCHER, VERY SIMPLE,

UPFUN: UPPER CORE FUNCTION DISPATCHER', ALSO SIMPLE,

CALLS DBPUT TO PUT IN WORD. IT THEN PUTS IN A 4000 TO END THE BUFFER.

DBPUT: THIS ROUTINE PUTS THE AC IN THE BUFFER. IF THE CONTENTS OF WHERE IT'S GOING=4001, THEN WE'VE RUN OUT OF ROOM AND A "TOO-BIG" MESSAGE IS GIVEN. IT RETURNS TO CALLER IF SUCCESSFULL.

UUDATA: THIS GIVES THE "A-D FULL" MESSAGE.

SETUP: THIS IS ORIGINALLY EDUSYS-10 CODE WHICH WAS MOVED BECAUSE WE NEEDED ROOM ELSEWHERE.

ISET: THIS ROUTINE INITIALIZES THE DISPLAY.

MEVAL: THIS ROUTINE EVALUATES AN EXPRESSION.
LEADING COMMAS ARE IGNORED, BAD SYNTAX GETS AN ERROR, RETURN IS TO CALLER WITH RESULTS IN FAC.

LLLJMS: THIS ROUTINE PROVIDES COMMUNICATION BETWEEN THE UPPER AND LOWER BANKS. TYPICAL CALL FROM UPPER IS

```
JMS     LLLJMS
YYYY
```

WHERE YYYY IS THE ROUTINE THE USER WISHES TO CALL IN LOWER BANK
RETURN IS ALWAYS TO UPPER FIELD.

LLLJMP: SAME AS LLLJMS EXCEPT THAT LLLJMS JMS'S AND LLLJMP JMP'S.

JUNOAD: GIVES THE "NO A-D" MESSAGE.

CCINTK: THIS ROUTINE CHECKS THE CLOCK FLAG. IF UP, THEN IT GOES TO CLOCK1 IN FIELD 1.

TST: DESCRIBED PREVIOUSLY.

- - - - - -

F I E L D 1

CLOCK1: THIS ROUTINE IS EXECUTED ON A CLOCK INTERUPT.
IT FIRST CLEARS THE FLAG AND SAVES THE STATUS IN CLKSTS.
IT THEN INCREMENTS THE TIME OF DAY [TIM2 AND TIM1].
THEN CHECKS ADACPT AND ABDGET TO SEE IF WE'RE SAMPLING.
IF WE ARE, THEN IT PICKS UP THE CHANNEL [ADCX] AND CALLS DOAD TO SAMPLE AS MANY TIMES AS NECESSARY AS DETERMINED BY ADCONT. IT CALLS APUT TO PLACE THE SAMPLES IN THE BUFFER. IT THEN INCREMENTS CT3,CT2 AND/OR CT1 TO FIND OUT WHETHER OR NOT DONE. IF DONE, IT CLEARS ADACPT AND SETS ABDGET MINUS.
IT NOW CHECKS TO SEE IF THE CLOCK IS UP AGAIN. IF IT IS , WE GET A RATE ERROR. IF IT ISN'T, WE CHECK FOR ^C TO SEE IF IN THE REGION WHERE WE ARE INTERUPTING SO FAST THAT WE CANNOT CHECK FOR A ^C IN THE TTY ROUTINES, BUT NOT FAST ENOUGH SO THAT THE CLOCK FLAG TEST WILL CATCH IT. THIS CONDITION

DDVL. IF INPUT IS FROM THE TELETYPE, ALL IS AS BEFORE.

FPT: THE FLOATING POINT PACKAGE IS AS BEFORE, EXCEPT FOR SOME CDF'S. THE ONLY TRICKY AREA IS THAT INDIRECTS GO THROUGH FIELD ONE, SO USE CAUTION.

TAB: DESCRIBED PREVIOUSLY (AS WITH TABDO).

INTER: THIS IS THE MAIN INTERUPT PROCESSOR. IT IS DESCRIBED ABOVE. THE ONLY THING NOT MENTIONED WAS BIDLE.

BIDLE: BIDLE IS A ROUTINE TO BE CALLED WHEN YOU WANT TO PUT BASIC TO SLEEP. ESSENTIALLY THIS SETS A FLAG CALLED BUSY. WHEN BASIC GOES TO EXIT FROM AN INTERUPT,
IT CHECKS THE STATUS OF BUSY. IF IT'S ZERO, IT EXITS NORMALLY. THIS MEANS IT WAS IN BASIC WHEN IT INTERUPTED. IF IT'S ONE, THEN IT WAS IN THE NULL JOB ROUTINE. IF SO, THEN NULLJOB COULD ONLY BE STARTED BY BIDLE, HENCE AFTER SERVICING THE INTEPUPT, IT RETURNS FROM BIDLE AND CLEARS THE BUSY FLAG. WHEN BIDLE IS CALLED AGAIN, BIDLE WILL EXIT TO THE LOCATION AS DEFINED BY WHAT THE INTERUPT ROUTINES SAVED WHEN IN THE LAST BUSY 1 STATE. HENCE BUSY IS THE WORD WHICH KEEPS TRACK OF WHAT IS GOING ON.

INTEXT: THIS ROUTINE EXITS FROM AN INTERUPT. IT INTEROGATES 'BUSY' TO FIND OUT WHAT TO DO.

PUTER: DESCRIBED PREVIOUSLY.

OBOP: POPS UP EITHER INPUT OR OUTPUT BUFFER FLAG.

RESET1: THIS RESETS ALL DEVICES. SEE PRESET.

RESET2: THIS IS A MASTER RESET OF ALL HARDWARE STATUS FLAGS. ONLY CALLED WHEN STARTING UP BASIC.

OUTCHL: DESCRIBED ABOVE.

DEVCOM: GETS NEXT ITEM FROM USER STATEMENT, THEN CALLS DEVCON.

DEVCON: CHECKS TO SEE IF ITEM IS A C.R. IF NOT, THEN A SYNTAX ERROR RESULTS.

RTER: THIS GIVES THE "RATE ERROR" MESSAGE.

GETARY: THIS ROUTINE LOOKS AT USER'S VARIABLE ARGUMENT. IT THEN PICKS UP SUBSCRIPTING INFORMATION. IT THEN CALLS GETADD TO RETURN LAST ELEMENT IN ARRAY. IT HAS THE FIRST, AND NOW THE LAST, SO IT HAS THE SIZE. IT LEAVES POINTER TO FIRST IN AC1 AND POINTER TO LAST IN AC2. THEN IT RETURNS.

PLOTB: THIS SETS UP DISPLAY BUFFER. IT SETS UP DISB TO POINT TO CORRECT LOCATION. IT PUTS A 4001 IN LAST WORD OF BUFFER, IT PUTS A 4000 IN FIRST WORD OF BUFFER SO NOTHING WILL BE DISPLAYED. THEN IT RETURNS.

DBLIT: THIS ROUTINE TAKES THE FAC AND STICKS IT IN THE BUFFER. IT FIRST CHECKS TO SEE IF IN RANGE (0 TO .9999999999) AND THEN IF NOT IT SETS DBAD. IT THEN FIXES FAC, AND THEN

II) NEW OR CHANGED ROUTINES

CLRCNT: THIS ROUTINE IS DESCRIBED BY CNCLR ABOVE.

PUTCH: THIS ROUTINE WAS CHANGED SLIGHTLY SO THAT IF PUTXRA WAS SET, A CERTAIN NUMBER OF NULL (ØØØ) CHARACTERS WOULD BE PUNCHED AFTER THE CRLF. THE NUMBER OF NULL'S PUNCHED IS A FUNCTION OF THE NUMBER OF CHARACTERS ON THE LINE. ALL OTHER FUNCTIONS REMAIN THE SAME.

OBLOW: WHILE NOT A ROUTINE, THIS IS WHERE THE RING BUFFER FOR
OBHIGH: THE OUTPUT DEVICE IS LOCATED. RIGHT NOW IT IS ABOUT 8 LOCATIONS LONG. IT EXTENDS FROM OBLOW TO OBHIGH.

PUTJ: THIS IS THE "PUT" FUNCTION. IMPLEMENTATION IS DESCRIBED IN THE I-O SECTION, ALONG WITH THE FUNCTION GET.

GETJ: SEE ABOVE.

PRINT: THIS ROUTINE HAS CHANGED SOMEWHAT. THE FUNCTION "CHECKW" HAS BEEN INCORPORATED TO SEE WHETHER OR NOT SOMETHING WILL FIT ON THIS LINE.

CHECKW: THIS FUNCTION IS CALLED WITH THE NUMBER OF PLACES YOU DESIRE TO PRINT IN THE AC. ACTUALLY, THE AC IS # OF PLACES DESIRED MINUS 1 (N-1). IF NOT ENOUGH ROOM, AN AUTOMATIC CRLF IS GIVEN. PRINT ALSO CHECKS THE SIZE OF THE LINE ON THE OUTPUT DEVICE. SIZE IS LEFT IN TWIDTH.

LIST: LIST NOW CHECKS TO SEE IF A * WAS GIVEN. IF SO, IT SETS THE PUTXRA FLAG BY CALLING POINT TO WAIT FOR I/O TO TERMINATE. SEE DESCRIPTION ABOVE.

BFIX: THIS ROUTINE WAS ADDED TO ENABLE THE EASIER CONVERSION OF INTEGER NUMBERS TO FAC NUMBERS. BEGFIX DOES THE FOLLOWING: IT CLEARS AC1 & AC2, IT SETS THE SIGN TO POSITIVE, IT SETS THE EXPONENT TO 233, WHICH IS CORRECT FOR FIXING A NUMBER, IT ALSO CLEARS THE OVERFLOW FLAG (OV), IT THEN RETURNS TO THE CALLER.

INPUT: THIS ROUTINE NOW CHECKS TO SEE IF THE INPUT IS FROM THE READER. IF IT IS, NO QUESTION MARK (?) IS GIVEN AND NO ECHO OF DATA

IMPORTANT ROUTINES AND LOCATIONS.

I) PAGE ZERO LOCATIONS.

0-2 USED FOR INTERUPT VECTOR.

DISAUTO, AJ10 REGISTER USED AS POINTER TO NEXT ITEM TO BE DISPLAYED (EITHER AN X OR Y POINT). FOUND IN NULJOB.

INDEV, CONTAINS INPUT DEVICE NUMBER (1=TTY, 2=PTR)

OUTD2, CONTAINS OUTPUT DEVICE WHILE ACCEPTING AND ECHOING INPUT FROM THE TTY, IN OTHER WORDS, A TEMPORARY ASSIGNED JUST TO HOLD THE 'REAL' OUTPUT DEVICE.

OUTDEV, OUTPUT DEVICE THE ROUTINES WILL USE (0=NOTHING, 1=TTY, 2=PTP, 3=LPT)

ODEV, OUTPUT DEVICE CURRENTLY BEING USED BY THE BUFFER.

CNTLC, THE CONTROL O FLAG (+0). IF 0, NO CONTROL O.

NOINT, A LOCATION WHICH TELLS WHETHER OR NOT THE ^C CHARACTER CAN INTERUPT BASIC. CERTAIN PROCESSES (SUCH AS CORE JUGGLING) CANNOT BE INTERUPTTED. THEREFORE THEY SET THE NOINT SWITCH TO A 0001. WHEN DONE, THEY RESET IT TO A 0. IF A ^C SHOULD BE STRUCK, IT WILL NOT STOP BASIC, BUT IT WILL SET THE NOINT SWITCH TO A 7777. WHEN WE LEAVE THE CRITICAL PROCESS, NOINT WILL BE CHECKED TO SEE IF IT CONTAINS A 7777, AND IF SO, WE PROCEED AS IF A CONTROL C HAD JUST BEEN STRUCK.

RBSWCH, THIS IS THE RUBOUTS SWITCH. 0=IGNORE RUBOUTS, <> TREAT RUBOUTS LIKE BACK-ARROWS (←), WHICH IS SHIFT O.

DISB, ADDRESS OF PRESENT DISPLAY BUFFER. IF NONE IS PRESENT, THIS LOCATION IS ZERO. BUFFER IS ALWAYS IN UPPER CORE.

PRESET, THIS POINTS TO A ROUTINE WHICH RESETS ALL FLAGS BACK TO TTY INPUT AND OUTPUT AND CLEANS UP THE BUFFERS, NECESSARY AFTER AN ERROR OCCURS.

SPECINT, THIS POINTS TO SUBROUTINE WHICH CHECKS FOR ADDITIONAL DEVICE INTERUPTS BESIDES THE STANDARD ONES. THE CLOCK TEST IS LOCATED THERE.

PCOAT, THIS POINTER POINTS TO A ROUTINE WHICH WAITS FOR OUTPUT TO TERMINATE (THE BUFFER EMPTY). THIS IS NECESSARY BECAUSE OTHERWISE ERROR AND/OR LISTING CONTROL INFORMATION MIGHT BE LOST IF A CONTROL O IS HIT AT THE WRONG TIME. IT ALSO RESETS THE CONTROL O FLAG. THIS ROUTINE IS ALSO SNEAKILY USED BY THE LIST * OPTION TO SET THE NULL AFTER CARRIAGE RETURN FLAG [PUTXRA], IT STORES THE AC IN PUTXRA FIRST, THEN WAITS FOR OUTPUT TO TERMINATE. HENCE THIS IS A TWO FOLD ROUTINE.

DELOUT, THIS POINTS TO A ROUTINE WHICH DELETES THE CURRENT CONTENTS OF THE OUTPUT BUFFER, USED FREQUENTLY BY MANY ROUTINES.

CNCLR, THIS ROUTINE TESTS AND CLEARS THE NOINT FLAG DISCUSSED EARILER.

SPLEFT, THIS POINTER POINTS TO SUBROUTINE WHICH DETERMINES WHETHER OR NOT A GIVEN OPERATION WILL OVERFLOW FREE CORE. THE AMOUNT OF ROOM YOU WISH TO TAKE SHOULD BE IN THE AC. IF NO ROOM, THEN AN ERROR MESSAGE IS GIVEN.

M) THE WAIT COMMAND.
THIS IS INTERNALLY IDENTICAL TO THE WAITC COMMAND, IT "FUDGES" THE TIME [TIM2] SO THAT WHEN AN INTERUPT OCCURS, IT LOOKS LIKE THE TIME HAS RUN OUT AND RETURNS.

N) THE ACCEPT COMMAND.
THE ACCEPT COMMAND WILL SET THE ACCEPT DATA SWITCH [ADACPT] IF AND ONLY IF WE HAVE A GOOD, CLEAN BUFFER [ABDGET>0], IF WE DO NOT, WE FALL THROUGH TO THE 'REJECT' COMMAND.

O) THE REJECT COMMAND.
THE REJECT COMMAND CLEARS THE ACCEPT DATA FLAG [ADACPT], IT RETURNS TO BASIC VIA DEVCOM.

P) THE REAL TIME COMMAND.
THIS COMMAND IS ONE OF THE MORE HAIRY OF THE NEW LAB-8/E BASIC COMMANDS. THE FOLLOWING IS WHAT OCCURS, IT FIRST CHECKS TO SEE IF WE ARE ACCEPTING DATA, IF WE ARE, THE COMMAND IS IGNORED [VIA SKIPIT], IF WE ARE NOT ACCEPTING DATA, WE CALL GETARY TO GET US THE AREA THE USER REQUESTED. THIS DONE, WE THEN USE THIS INFORMATION TO SET THE BUFFER POINTERS [APUT1 AND APUT2] AND THE BUFFER LIMIT CONTROL INFORMATION [ADA1, ADA2, AND ADA3]. WE THEN GET THE CHANNEL NUMBER TO START [VIA UMEVAL] AND PLACE IT IN ADCX, WE THEN GET THE NUMBER OF CHANNELS TO USE AND PLACE IT IN ADCUNT, WE FINALLY GET THE NUMBER OF TIME COUNTS TO DO AND LEAVE IT IN CT1, CT2 AND CT3. REMEMBER THAT THE TIME IS DOUBLE WORD, HENCE THE DOUBLE WORD ARITHMETIC AT THIS POINT, CT1 IS USED TO GIVE US A THREE WORD ZERO COUNTER SHOULD THE USER PUT IN 0 AS NUMBER OF COUNTS. NOTE THAT 2^36 IS A VERY LONG TIME. WE THEN ZERO OUT THE NUMBER OF SAMPLES PRESENTLY IN THE BUFFER [ACOUNT] AND TELL THE SYSTEM THAT WE'VE GOT A GOOD BUFFER [ABDGET >0].

Q) THE TIME FUNCTION [TIM]
THE TIME FUNCTION RETURNS THE NUMBER OF TICS, EVERY TIME THE CLOCK INTERUPTS, CLOCKI (THE CLOCK INTERUPT ROUTINE) INCREMENTS TIM2 AND THEN TIM1. TIM RETURNS THIS TIME IN THE AC.

R) THE ADB FUNCTION.
THIS FUNCTION CALLS UADCB WITH THE AC ALL 7777'S. UADCB WILL BE DESCRIBED NOW. UADCB IS THE MASTER A-D FUNCTION SWITCHER. UADCB EITHER DOES AN IMMEDIATE ADC IF THE AC IS ZERO OR WILL ATTEMPT TO GET A SAMPLE FROM THE USER'S A-D BUFFER IF THE AC IS NON-ZERO. IF THE AC IS ZERO, IT WILL CALL DOAD WITH THE INTEGERIZED AC3 (BITS 9-11). DOAD DOES AN A-D CONVERSION ON THE CHANNEL IN THE AC. IT THEN RETURNS THIS VALUE IN THE AC. UADCB THEN GOES TO UINAC, WHERE THE A-D VALUE IS PLACED IN THE FAC, NORMALIZED AND THEN BROUGHT INTO THE CORRECT NUMERIC RANGE (-1.V TO +1.V), THE FUNCTION THEN RETURNS. IF THE AC IS NOT ZERO (FOR AN ADB), IT CHECKS TO SEE IF THE BUFFER IS ACTIVE, IF NOT, AN ERROR IS GIVEN. IT THEN CHECKS TO SEE IF THE BUFFER IS ACTIVE, BUT THE TIME HAS RUN OUT. IF SO, ANOTHER MESSAGE IS GIVEN, IF ALL IS WELL, IT CALLS AGET TO GET AN A-D VALUE FROM THE USERS BUFFER, AGET IS MERELY THE RING BUFFER REMOVER, IF NOTHING IS IN THE BUFFER [ACOUNT=0], THEN AGET WILL PUT BASIC TO SLEEP. AGET RETURNS WITH CONVERTED NUMBER IN AC, UADCB THEN GOES TO UINAC WHICH WAS DESCRIBED ABOVE.

HEAD TO THE DESIRED POSITION. IT CONSIST OF TWO PARTS.
THE FIRST IS THE ACTUAL TAB FUNCTION. THE SECOND IS CALLED
'TABDO' AND IS CALLED BY PRINT AFTER EVALUATING AN EXPRESSION
IF THE TABFLG IS SET. WHAT HAPPENS IS AS FOLLOWS:
WHEN TAB IS CALLED, IT FIRST FIXES THE ARGUMENT. IT THEN SETS
'TABFLG' TO INDICATE THAT WE HAVE PROCESSED A TAB FUNCTION.
IT THEN GETS THE PRESENT POSITION OF THE PRINT HEAD FROM
'COLUMN' AND RETURNS THIS AS THE FUNCTION VALUE. THUS TAB CAN
BE USED AS A REGUALR FUNCTION. IT THEN RETURNS TO THE CALLER.
THE PRINT STATEMENT PROCESSOR CHECKS TABFLG AFTER EVALUATING AN
EXPRESSION. IF IT'S SET (NON-ZERO) THEN IT GOES TO TABDO. TABDO
PICKS UP WHERE WE WANT TO GO TO (LEFT IN 'TABDES' BY TAB) AND
FIGURES OUT WHERE WE ARE BY USING COLUMN. IF WE ARE TO FAR IT
GIVES A CARRIAGE RETURN AND A NULL CHARACTER(TO PREVENT
TIMING PROBLEMS). IT THEN SPACES OVER THE CORRECT NUMBER OF
SPACES. WHEN DONE IT RETURNS TO 'PRINT+1' TO
FINISH PROCESS THE COMMAND. NOTE THAT WE CANNOT RETURN TO THE
SECTION WHICH CALLED IT BECAUSE IT WILL PRINT OUT A VALUE (
SUCH AS 23) BECAUSE IT WILL TAKE THE EVALUATED NUMBER AND
CONVERT IT TO ASCII.

THE FOLLOWING COMMANDS ARE ALL DONE IN (RESIDE) IN FIELD 1.

THE ADC FUNCTION.
THIS FUNCTION [RENAMED ZZADC] CALLS THE ROUTINE UADCB WITH
A ZERO AC. UADCB IS THE GENERAL A-D
DISPATCHER. UADCB HAS TWO COURSES OF ACTION DEPENDING ON THE AC.
THE USER SHOULD READ IT'S DESCRIPTION AT
THE END. UADCB WILL RETURN WITH THE CORRECT ADC VALUE IN THE
FAC. IT WILL THEN RETURN TO THE CALLER.

THE SET RATE COMMAND.
THIS COMMAND SETS THE CLOCK GOING FROM THE ARGUMENTS. IT
FIRST CALLS MEVAL [VIA UMEVAL] TO GET THE RATE (0-7). IT THEN
PLACES THIS IN THE COMMAND REGISTER ALONG WITH A 5010. THESE ARE
THE CORRECT HARDWARE FLAG BITS TO RUN IN 'NORMAL' CLOCK MODE.
IT THEN JUMPS TO USETM. USETM WILL EVALUATE THE NEXT EXPRESSION
[VIA UMEVAL] AND WILL THEN LOAD THE CLOCK REGISTER
WITH THIS NUMBER. IT THEN CLEARS THE TWO TIME REGISTERS (TIM1
AND TIM2). THESE ARE INCREMENTED EVERY TIME A CLOCK TIC OCCURS.
IT THEN RETURNS TO THE CALLER.

THE SET CLOCK COMMAND.
THIS IS SIMIALIAR TO THE SET RATE COMMAND EXCEPT THAT THE
ENTIRE 12 BITS OF THE MODE THE USER HAS SPECIFIED
IS USED. JUST BIT 8 IS SET ON TO INSURE INTERUPTS. THEN IT GOES
TO USETM.

THE WAITC COMMAND.
THIS COMMAND WAITS FOR A CLOCK TIC. IT FIRST PICKS UP THE LOW
ORDER WORD OF THE CLOCK COUNTER [TIM2] AND SAVES IT IN UTEMP.
IT THEN CALLS BIDLE TO PUT BASIC TO SLEEP. WHEN IT'S AWAKENED.
IT SEES IF THE TIME HAS CHANGED. IF IT HAS, IT RETURNS
VIA DEVCOM. IF IT HASN'T, IT GOES TO SLEEP AGAIN.

A) THE NULL JOB (OR NULJOB). THIS ROUTINE IS CALLED WHENEVER BASIC IS WAITING FOR SOMETHING TO HAPPEN. IT RUNS THE SCOPE IN LAB-8/E BASIC. IT IS ALSO CALLED VIA THE "DELAY" COMMAND. THE DIFFERENCE IS JUST IN THE MANNER OF EXIT. THE FLAG 'NDELAY' CONTROLS THAT. NULJOB CHECKS TO SEE IF THERE IS A PLOTTING BUFFER ASSIGNED. IF THERE IS (DISB IS NON-ZERO) THEN IT WILL SET UP DISAUTO TO BE DISB. IT WILL THEN TAKE OUT X-Y PAIRS OF POINTS AND DISPLAY THEM. A MINUS ENTRY IS THE END OF LIST INDICATOR. ON END OF LIST, IT GOES BACK TO NULJOB AND CHECKS NDELAY TO SEE IF IT SHOULD RETURN TO THE CALLER. IF NOT, IT THEN RESTARTS ALL OVER AGAIN BY RECHECKING DISB.

B) THE PLOT STATEMENT.
WHENEVER A PLOT COMMAND IS ENCOUNTERED, BASIC GOES TO 'PLOT'. PLOT FIRST CHECKS TO SEE IF A BUFFER IS ASSIGNED. IF ONE IS NOT ASSIGNED, IT ASSIGNS SPACE BY CHANGING ARRLOC. IT THEN CALLS PLOTB TO SET UP THE CORRECT BUFFER WORDS. WHEN A BUFFER IS PRESENT, PLOT CALLS MEVAL. MEVAL EVALUATES AN EXPRESSION. IT THEN MULTIPLIES THIS EXPRESSION BY FSHIFT FOR CORRECT SCREEN SCALING. IT THEN CALLS DBLIT TO PLACE IN THE SCOPE BUFFER. IT THEN GOES THE SAME FOR THE Y VALUE (EXCEPT NOT MULTIPLYING). WHEN ALL IS DONE, IT THEN GOES TO DEVCON TO CLEAN UP THE COMMAND.

C) THE USE STATEMENT.
THIS STATEMENT ALLOCATES A SCOPE BUFFER IN BASIC. IT FIRST CHECKS TO SEE IF A BUFFER IS ASSIGNED. IF ONE IS, IT RETURNS. IF ONE ISN'T, THEN IT CALLS GETARY TO FIGURE OUT THE SIZE OF THE ARGUEMENT AND TO ALLOCATE THE SPACE (ACTUALLY, IT'S ALREADY ALOCATED). IT THEN SETS DISB TO THE CORRECT ADDRESS AND THEN CALLS PLOTB TO SET UP BUFFER. IT THEN EXITS TO DEVCON.

D) THE CLEAR COMMAND.
THIS STATEMENT CAUSES THE SCOPE BUFFER TO BE CLEARED. IF THERE IS A BUFFER PRESENT (DISB<>0), THEN IT PLACES AN ABORT CODE (4000) AS THE FIRST INSTRUCTION IN THE BUFFER. NULJOB WILL STOP DISPLAYING ON THIS, HENCE NOTHING WILL BE DISPLAYED.

E) THE TST FUNCTION.
THE TST FUNCTION TESTS WHETHER OR NOT A CHARACTER HAS BEEN TYPED BY PLACING 'INCHAR' IN AC3 AND THEN NORMAL ZING IT. BECAUSE INCHAR IS ALWAYS ZERO IF NO CHARACTER HAS BEEN TYPED AND IS ALWAYS NON-ZERO IF A CHARACTER HAS BEEN TYPED. THIS MAKES TST BEHAVE AS DESIRED AND DESCRIBED.

F) THE GET AND PUT FUNCTIONS.
THE GET AND PUT FUNCTIONS PERFORM THERE TASKS BY MERELY CALLING PUTCH AND GETCH WITH THE CORRECT ITEM IN THE AC. THEY INSERT-REMOVE THE 8 BIT CHARACTER FROM AC3. FAIRLY TRIVIAL ROUTINES.

G) THE RUBOUTS AND NO RUBOUTS COMMANDS.
THESE COMMANDS SET THE VARIABLE RBSWCH TO A 1 OR 0 DEPENDING ON WHETHER OR NOT TO PROCESS RUBOUTS. THE ROUTINE GETCH TESTS THIS FLAG WHEN INPUTING CHARACTERS.

H) THE TAB FUNCTION.
THIS FUNCTION IS THE FUNCTION WHICH CAN POSITION THE PRINT

THEREFORE, IT IS POSSIBLE TO STORE AWAY CHARACTERS IN IT'S BUFFER TO BE OUTPUT LATER. THE BUFFER IS LOCATED AT OBLOW TO OBHIGH. THEREFORE, TO PREVENT DEVICE MIX-UPS, PUTER HAS IT'S OWN OUTPUT FLAG. IT IS CALLED 'ODEV'. ODEV CONTAINS THE NUMBER OF THE DEVICE WHICH IS NOW USING THE BUFFER. SHOULD PUTER BE CALLED TO PRINT A CHARACTER ON A DEVICE THAT IS DIFFERENT FROM THE ONE WHICH IS NOW USING THE BUFFER, PUTER WILL WAIT (VIA BIOLE) UNTIL THE BUFFER IS EMPTY, THEN IT WILL CONTINUE. THERE ARE TWO OTHER POSSIBLITIES. ONE IS THAT THE BUFFER IS EMPTY, THE OTHER IS THAT THERE ARE CHARACTERS IN IT BUT GOING TO THE SAME DEVICE. IF THE BUFFER IS EMPTY, PUTER 'ASSIGNS' THE BUFFER TO THE DEVICE AND CALLS 'OUTIT' TO OUTPUT THE CHARACTER. OUTIT REMOVES ONE CHARACTER FROM THE BUFFER AND OUTPUTS IT, IF IT'S POSSIBLE. IF THERE ARE CHARACTERS IN THE BUFFER, THEN PUTER MERELY STICKS THE CHARACTER IN THE BUFFER AND RETURNS. THERE IS ONE OTHER FLAG ASSOCIATED WITH THE BUFFER THAT SHOULD CONCERN THE READER, IT IS CALLED 'CNTLO' AND IS THE CONTROL O FLAG. THIS FLAG ESSENTIALLY SAYS WHETHER OR NOT WE ARE UNDER A CONTROL O (SUPPRESS OUTPUT). IF WE ARE INDEED UNDER A CONTROL FLAG, THEN WE DO NOT PROCESS CHARACTERS. INSTEAD WE JUST RETURN IMMEDIATELY.

C) THE INTERUPT CHAIN:

THIS IS THE ROUTINE WHICH CHECKS THE DEVICES ON AN INTERUPT. IT BASICALLY DOES THE FOLLOWING. IT FIRST SAVES THE STATE OF THE MACHINE (AC, LINK, ETC.). IT THEN DOES IOT'S TO FIND OUT WHICH DEVICE IS INTERUPTING. IF IT CANNOT FIND IT, IT HALTS (A FATAL ERROR). IF IT DOES FIND IT, IT GOES TO THE CORRECT ROUTINE. ON AN OUTPUT FLAG (TTY,LPT, PTP) IT CLEARS THE FLAG AND THEN CALLS OUTIT TO PRINT THE NEXT CHARACTER. ON A HIGH SPEED READER INTERUPT, IT PUTS THE CHARACTER IN HRCHAR. ON TTY INTERUPT, IT CHECKS TO SEE IF IT'S A CONTROL CHARACTER (↑C OR ↑O). IF IT ISN'T, IT MERELY PLACES THE CHARACTER IN INCHAR. IF IT IS A ↑O, THEN IT SETS THE NTLO FLAG AND CALLS 'OUTDEL' TO DELETE THE RESENT OUTPUT BUFFER. IF IT'S A ↑C, IT CHECKS NOINT (DISCUSSED AT HE END). IF IT'S OK TO STOP BASIC NOW, IT THEN RESETS ASIC WITH A STOP-READY MESSAGE AND GOES TO EDIT (THE EDITOR PORTION). HE ONLY OTHER DEVICE IN THE INTERUPT CHAIN IS THE CLOCK, WHICH WILL E DISCUSSED LATER. TO EXIT FROM AN INTERUPT, 'INTEXT' RESTORES THE ACHINE (AC, LINE, ETC.)

I) LAB-8\E ADDITIONS:

THE LAB-8\E ADDITIONS ARE BASICALLY A SERIES OF COMMANDS WHICH ERATE ON THE LAB-8\E PERIPHERALS. SINCE THE READER IS FAMILIAR WITH USYS-10, THE ACTUAL IMPLEMENTAION WILL NOT BE DESCRIBED, JUST THEIR LATIONSHIP WITH THE REST OF BASIC WILL BE DESCRIBED.

IS PROCESSED, OR WHEN ONE IS TYPED, THE I-O ROUTINES WILL RETURN TO BASIC AND BASIC WILL CONTINUE TO 'RUN' UNTIL IT WANTS ANOTHER CHARACTER OR UNITL IT WANTS TO OUTPUT ANOTHER CHARACTER. IN REALITY, BASIC IS USUALLY WAITING FOR INPUT, WHEN BASIC CALLS THE I-O ROUTINES, THE I-O ROUTINES CHECK TO SEE IF IT CAN DO WHAT BASIC WANTS. IF IT CAN, IT DOES IT AND RETURNS TO BASIC. IF IT CANNOT, THEN IT WILL WAIT FOR AN INTERUPT TO OCCUR AND SEE IF CAN PROCESS IT. IF IT STILL CANNOT, IT WILL WAIT AGAIN UNTIL THE DESIRED CONDTION IS MET. WHILE WAITING, THE I-O ROUTINES 'PUT BASIC TO SLEEP'. THEY DO THIS BY EXECUTING A LITTLE PROGRAM CALLED NULJOB. NULJOB IS RUN WHENEVER THE I-O ROUTINES ARE WAITING FOR AN INTERUPT. IN LAB-8/E BASIC, NULJOB IS THE ROUTINE WHICH DISPLAYS THE CONTENTS OF THE DISPLAY BUFFER.

BASIC CALLS THE I-O ROUTINES TO GET OR PUT A CHARACTER. THE USE OF THE SCOPE WILL BE DISCUSSED IN THE NEXT SECTION.

A) THE GET ROUTINE:

THIS ROUTINE REQUESTS THE I-O ROUTINES TO GET A CHARACTER. THERE IS A FLAG ASSOCIATED WITH THIS ROUTINE. THIS FLAG IS CALLED 'INDEV'. INDEV IS EITHER A 1 OR 2. IF IT'S A ONE, THE GET ROUTINE (CALLED GETCH) WILL GET A CHARACTER FROM THE TTY. IF IT'S A TWO, IT WILL GET IT FROM THE HIGH SPEED READER. THE SEQUENCE OF EVENTS IS AS FOLLOWS:

GETCH FIRST CHECKS THE DEVICE FLAG. IF IT'S A ONE, THEN IT CHECKS TO SEE IF A CHARACTER HAS BEEN TYPED. TYPED CHARACTERS ARE LEFT BY THE INTERUPT ROUTINES IN A LOCATION CALLED 'INCHAR'. IF NO CHARACTER HAS BEEN TYPED, IT CALLS A ROUTINE CALLED 'BIDLE'. BIDLE PUTS BASIC TO SLEEP (RUNS THE NULL JOB) UNTIL AN INTERUPT OCCURS. AFTER BIDLE RETURNS, GETCH AGAIN LOOPS AND CHECKS INCHAR. IF STILL NOT SET, IT PUTS ITSELF TO SLEEP AGAIN.

FOR THE HIGH SPEED READER, THE SEQUENCE IS DIFFERENT. THE ROUTINE 'GWHERE' FIRST REQUESTS A CHARACTER FROM THE HIGH SPEED READER. IT THEN INCREMENTS A COUNTER AND TESTS THE WORD 'HRCHAR'. ON AN INTERUPT FROM THE HIGH SPEED READER, THE INTERUPT ROUTINES PLACE THE CHARACTER IN HRCHAR. IF GWHERE GETS A CHARACTER FROM HRCHAR BEFORE TIME RUNS OUT, IT JUMPS TO THE MIDDLE OF GETCH. IF TIME RUNS OUT, THEN IT PRINTS THE MESSAGE 'TTY' ON THE TTY, AND THEN RESETS INDEV TO 1, AND THEN JUMPS TO NEAR THE BEGINNING OF GETCH TO WAIT FOR A TTY CHARACTER.

ASSUME THAT WE ARE NOW IN GETCH WITH A CHARACTER. WE NOW 'MASK' ALL ALTMODES INTO 175. IF RUBOUTS ARE SELECTED, WE MASK 177'S INTO 137'S. BACKSPACE (210) IS ALSO MASKED INTO 137. THUS SOME PRE-EDITING IS DONE ON THE CHARACTER. IT IS THEN RETURNED IN THE AC FROM GETCH.

) THE PUT ROUTINE:

THIS ROUTINE PRINTS A CHARACTER ON THE DESIRED OUTPUT DEVICE. IT IS ALOT MORE COMPLEX THEN GET. LIKE GET, THERE IS A FLAG ASSOCIATED WITH THE OUTPUT DEVICE. IT IS CALLED 'OUTDEV'. OUTDEV RANGES FROM Ø TO 3. Ø IS NO OUTPUT, 1 IS TTY OUTPUT, 2 IS PTP OUTPUT, AND 3 IS LPT OUTPUT. 'PUTER', THE OUTPUT ROUTINE, IS FULLY BUFFERED.

17734-17777 LEFT ALONE [PRESERVED FOR THE RIM LOADER]

ALL OF THE SPACE THAT WAS IN FREE FIELD 0 IS NOW OCCUPIED BY CODE. THE BINARY LOADER MUST BE IN FIELD ONE DURING LOADING. WHEN BASIC IS STARTED, IT WILL DESTROY IT (BECAUSE IT IS IN THE BUFFER AREA), ONLY THE RIM LOADER WILL BE PRESERVED. LAB-8/E BASIC MAY NOT BE LOADED BY PS/8.

NOW THAT THE CORE LAYOUT HAS BEEN ESTABLISHED, THE MODE OF OPERATION WILL BE DISCUSSED. BASIC IN IT'S NORMAL MODE OF OPERATION WILL BE 'DIDDLING' WITH THE SYMBOL TABLE (EITHER THE PERMANENT ONE OR THE USER'S ONE), OR IT WILL BE EDITING, READING IN, OR TRANSLATING A LINE. THE LINE BUFFER IS IN FIELD ONE, AS IS THE SYMBOL TABLE, FOR-NEXT LIST, GO SUB LIST, ETC. THEREFORE, BASIC WILL BE SPENDING MOST OF IT'S TIME IN FIELD ONE. THIS MEANS THAT BASIC'S NORMAL MODE OF OPERATION IS FOR THE DATA FIELD TO BE SET TO FIELD 1. THEREFORE, THE DATA FIELD IS ALWAYS SET TO FIELD ONE EXCEPT IN THOSE RARE INSTANCES WHERE BASIC MUST REFERENCE FIELD 0 DATA INDIRECT. ONE SUCH PLACE WHERE THIS HAPPENS IS THE FLOATING POINT PACKAGE. THE FLOATING POINT PACKAGE SPENDS MOST OF IT'S TIME IN FIELD ZERO MODE. TO FIND OUT WHERE SUCH SPOTS OCCUR, THE CDF INSTRUCTIONS ARE TAGGED WITH THE COMMENT "/''''''''''' 8K INSERT", THUS IF THE READER IS MODIFYING ONE OF THESE SECTIONS, HE SHOULD BE CAREFUL OF PICKING UP INDIRECT DATA OR EXITING WITH THE DATA FIELD SET INCORRECTLY. ALL TOTALED, THERE ARE RELATIVELY FEW CDF INSTRUCTIONS FOR THE SIZE OF A PROGRAM LIKE BASIC.

II) INTERUPT I/O

WE NOW COME TO THE SECTION WHICH MAKES THE BIGGEST DIFFERENCE BETWEEN EDUSYS-10 AND LAB-8/E BASIC. INTERUPT I/O GIVES LAB-8/E BASIC A 'SIMULATED' BETTER RUN-TIME BECAUSE OF BUFFERING AND ALLOWS THE CLOCK TO BE USED IN A REAL-TIME MANNER. A BASIC OVERVIEW FOLLOWS:

IT WAS STATED IN THE PREVIOUS SECTION THAT BASIC SPENDS MOST OF ITS TIME IN FIELD ONE. THAT IS MOSTLY CORRECT, IF WE TAKE THE WORD 'BASIC' AS MEANING THAT PART WHICH RUNS OR TRANSLATES THE PROGRAM. THERE EXIST OTHER SECTIONS OF THE PROGRAM WHICH RUN WITHOUT EVER REFERENCING ANYTHING IN FIELD ONE. THE I-O ROUTINES ARE ONE OF THESE. THE I-O ROUTINES ARE BASIC'S WAY OF DEALING WITH THE OUTSIDE WORLD. IF BASIC WANTS TO PUT OR GET A CHARACTER, IT CALLS THE I-O ROUTINES TO DO THIS TASK. WHEN THE CHARACTER

LAB-8/E DOCUMENTATION

THIS IS A DESCRIPTION OF THE INTERNAL WORKINGS OF BASIC FOR THE LAB-8/E. SINCE LAB-8/E BASIC IS A MODIFICATION OF EDUSYS-10(OTHERWISE KNOWN AS 4K BASIC), THIS DOCUMENT WILL ONLY DESCRIBE THE CHANGES MADE TO EDUSYS-10 TO PRODUCE LAB-8/E BASIC.

LAB-8/E BASIC WORKS APPROXIMATELY THE SAME WAY AS EDUSYS-10 DOES. IT DIFFERS IN THE FOLLOWING RESPECTS:

1) LAB-8/E BASIC USES 8K OF CORE AS OPPOSED TO 4K OF CORE.
2) LAB-8/E BASIC USES INTERUPTS AND HANDLES MORE DEIVCES.
3) LAB-8/E BASIC HAS ADDITIONAL CODE TO HANDLE THE SPECIAL COMMANDS WHICH PERTAIN TO THE LAB-8/E.

WHAT FOLLOWS IS A LIST OF THE CHANGES MADE TO IMPLEMENT THE ABOVE FEATURES. KNOWLEDGE IS ASSUMED OF BASIC AND EDUSYS-10. A BRIEF DESCRIPTION IS PROVIDED AT THE END OF THIS MEMO WHICH DECRIBES WHAT THE ROUTINES ADDED DO, AND WHAT KEY CORE LOCATIONS MEAN.

1) 8K MODIFICATION.

TO MAKE BASIC USE 8K, THE FOLLOWING CHANGE WAS MADE. IN 4K EVERYTHING RESIDED IN FIELD 0. IN THE 8K VERSION, EVERYTHING FROM THE ARRAY SPACE ON UP NOW RESIDES IN FIELD 1. IN ADDITION, SOME CODE NOW RESIDES IN FIELD ONE TO HANDLE THE CLOCK AND A-D INSTRUCTIONS. THUS A CORE MAP WOULD LOOK LIKE:

0000-10777 BASIC SYSTEM CALL CODE NEEDED FOR BASIC]
11000-17755 BASIC SYMBOL AREA. THIS AREA IS THE SAME AS IN 4K BASIC: IT CONSISTS OF THE FOLLOWING IN THIS ORDER:

1) ARRAY AND VARIABLE SPACE
2) FREE SPACE (EVERYTHING 'GROWS' INTO THIS SPACE)
3) CODIFIED (COMPILED) BASIC PROGRAM IMAGE.
4) USER SYMBOL TABLE AREA,
5) BASIC'S PERMANENT SYMBOL TABLE AREA.
6) THE LINEBUFFER (TTY AND TRANSLATED LINE BUFFER)
7) THE STACK,
8) THE FOR-NEXT LIST
9) THE GO SUB LIST.

THUS UPPER CORE IS THE AREA WHERE ALL THE DRIVING TABLES AND PROGRAM-VARIABLE AREAS ARE IN BASIC.

digital INTEROFFICE MEMORANDUM

ROB KRUYK ONTVANGEN - 7 NOV. 1972
THE HAGUE

DATE: October 10, 1972

FROM: J. Duffy

DEPT: Small Systems

SUBJ: User functions and Commands in 8K Basic R/T

Before one tries to implement any functions or commands the following articles should be acquired.

1. Programming Languages 1972

2. Basic/RT User's Manual DEC-LB-U7ØB-D or the newer LAB8/e Manual, LAB8/e Software Systems User's Manual DEC-8E-ALUMA-A-D.

3. A Listing of Basic/RT DEC-8E-ABASA-A-LA.

```
FATNCJ,
LIMIT,  FIRST LOCATION OF USER FREE SPACE,
        'PLIMIT' IS THE ONLY POINTER TO IT SO THAT CHANGING 'PLIMIT'
        IS ENOUGH TO REMOVE THE FUNCTIONS, THE INITIAL DIALOG
        DOES THIS ON COMMAND.
PERMSYM,THIS IS THE HIGHEST LOCATION OF CHANGEABLE USER STORAGE.
        THE PERMANENT SYMBOL TABLE IS JUST ABOVE IT.

        ITS FORM IS EXACTLY WHAT IT LOOKS LIKE:
        CODE WORD;TEXT 'THE SYMBOLS PRINT NAME'

LINBUF, THE LINE INPUT BUFFER, TRANSLATED LINE STARTS HERE,
        THERE IS NO COUNTER FOR THE FULLNESS OF THE LINE EXCEPT WHEN IT
        IS JUST COLLECTED,
LBEGIN, MORE OF THE LINE INPUT BUFFER, UNTRANSLATED ASCII COMES IN
        STARTING HERE FROM THE TELETYPE,
ENDLIN, THE ASCII CAN EXTEND AS FAR AS 'ENDPDL', BUT THE TRANSLATED
        LINE MUST END AT 'ENDLIN',
PDLIST, PUSH DOWN LIST USED ONLY FOR EXPRESSIONS AND THINGS LIKE THAT,
        'PDL' POINTS INTO THIS,
ENDPDL, END OF THE PUSH DOWN LIST,
FORLIST,FOR TABLE, A MAXIMUM OF 8 TWO WORD ENTRIES,
        A VARIABLE, AND THE LOCATION OF THE 'TO' IN THE 'FOR' STATEMENT,
        'FORCT' IS THE ONES COMPLEMENT OF THE NUMBER OF ENTRIES,
GOLIST, GOSUB TABLE, A MAXIMUM OF 8 ENTRIES,
        A POINTER TO AFTER THE GOSUB,
        'GSBPTR' POINTS INTO HERE TO THE NEXT FREE SPACE,
GSBEND, END OF THE GOSUB TABLE,
        7756 THROUGH 7777 ARE UNTOUCHED BY EXECUTION OF BASIC,
```

```
FSI'.C,
FSI'OR,
PSG',
PINT,
FTA:T1,
FTA.T2,
TSIN2,
FSIN2Z,
FSINC1,
FSINC3,
FSINC4,
FSINC5,
FSINC6,
CSINC7,
CSINM4,
UPARPX, BASIC '*' USING EXTENDED FUNCTIONS.
FUPRC1, IF THE SECOND ARGUMENT IS SUITABLE, DO IT BY THE MULTIPLICATION
        METHOD, OTHERWISE,
EXPLON3,A*B = EXP(LOG(A)*B) WITH POSSIBLE NEGATIVE ARG, TO LOG,
PLOG,
PEXP,
EXP,    BASIC 'EXP' FUNCTION.(CAN BE DELETED.),
FXXPFX,
PPINT,
FEXPI,
FEXPU,
FEXPF,
FEXPC1,
FEXPC2,
FEXPC3,
FEXPC4,
FEXPC5,
FEXPC6,
LOG,    BASIC 'LOG' FUNCTION.(CAN BE DELETED.),
FLOGC2,
FLOGC3,
FLOGC4,
LOGFWD,
LOGACE,
LOGOK2,
FLOGC1,
ATN,    BASIC 'ATN' FUNCTION.(CAN BE DELETED.),
ATNLOW,
ATNNC1,
ATNBI',
FATN3',
FATNA',
FATNT,
FATNTT,
FATNC1,
FATNC2,
FATNC3,
FATNC4,
 ATNC5,
FATNC6,
FATNC7,
FATNC8,
FATNC9,
FATNC,
FATNCH,
```

```
        IT RETURNS WITH THE AC EQUAL TO THE LOW ORDER WORD PLUS
        (IF ZERO ORIGIN INDEXING, THEN 1, ELSE Ø).
FIXLOP,
ZFIXEX,
FIXLPIT,
SSFIX,
07545,
ARGERR, PRINT 'ARGUMENT ERROR'
DOTZERO,',Ø'
OUTNUM, ROUTINETO PRINT FLOATING AC IN BASIC FORMAT. GIVES NO LEADING
        OR TRAILING SPACES.
NONZERO,
CVTLOOP,
FMT2,
FMT1,   THIS ENTIRE ROUTINE IS STOLEN FROM TSS8 BASIC.
FMT3,
TRYAGIN,
ZEROONE,
DIGLOP,
FIXUP,
 17,    LITERALS,
 253,
0255,
0305,
07525,
PMPY,
PNEPE,
FIXITUP,
DCC,
IORMIT,
IORMED,
 OTX12,
 XPGOOD,
 VLOOP,
 XPOK,
MPY,
NUMBUF, BUFFER FOR HOLDING ACCUMULATED NUMBER BEFORE PRINTOUT.
02062,  LITERALS,
07613,
07773,
RND,    BASIC 'RND' FUNCTION.(CAN BE DELETED.).
ERND2,
TRNDX,
TRNDC1,
FSQRX,
SQR,    BASIC 'SQR' FUNCTION.(CAN BE DELETED.).
SQLOOP,
SQEXIT,
SQEXIT,
07672,
FN,     BASIC 'FN@' FUNCTION.(CAN BE DELETED.).
        EXPECTS 'USERFN' TO HAVVE A POINTER TO A LINE STARTING 'DEF',
        IT CHECKS FOR THAT, THEN 'PUSH'ES THE VALUE OF THE FORMAL
        PARAMETER, SETS IT EQUAL TO THE ACTUAL PARAMETER,
        EVALUATES THE RIGHT SIDE OF THE 'DEF' STATEMENT,
        THEN 'POP'S THE OLD VALUE OF THE FORMAL PARAMETER.
PUSHFN,
TAN,    BASIC 'TAN' FUNCTION.CAN BE DELETED.).
COS,    BASIC 'COS' FUNCTION.(CAN BE DELETED.).
SIN,    BASIC 'SIN' FUNCTION.(CAN BE DELETED.).
```

```
LOADED, RETURN WITH IT IN THE FLOATING AC.
ISLIT,  SPACE SAVER IN ISITLIT.
ERROR,  ERROR MESSAGE PRINTING ROUTINE.
QERROR, 'ERROR'
FPT,    SLIGHTLY ABNORMAL FLOATING POINT PACKAGE- 27 BIT MANTISSA.
        HAS 7 ADDRESSABLE INSTRUCTIONS;
        FLOATING JUMP,
        FLOATING STORE,
        FLOATING LOAD,
        FLOATING ADD,
        FLOATING SUBTRACT,
        FLOATING MULTIPLY,
        AND FLOATING DIVIDE.
        ADDRESSING IS IS PAGE ZERO DIRECT OR INDIRECT IF BIT 4 IS OFF.
        OTHERWISE IT IS RELATIVE TO THE PROGRAM COUNTER.
        THERE ARE ALSO THE SIX FLOATING POINT SKIPS, AND AN EXIT.
'PLOOP, NOW YOU COME TO MY(L.W.E.) BASIC PHILOSOPHY ON DOCUMENTATION.
PPGE,   IF THE READER DOES NOT KNOW PDP-8 CODE WELL ENOUGH TO READ THE
PDOIT,  LISTING, HE SHOULD NOT BE MODIFYING THE CODE. THIS IS ESPECIALLY
PNOADR, TRUE FOR NUMBER CRUNCHING.
PGOTO,
PJUMP,
PADDR,
POPER,
PSKIP,
PJMP
PSTO
IGHWD,
'600
PLAC,
PARR2,  MORE OF BASIC '**' EXPONENTIATION. IF THE SECOND ARGUMENT
        WAS SUITABLE, IT IS DONE BY REPEATED MULTIPLICATION.
R1,
CN,
)P3,
ADD,
PSUB,
PADD
LGNLI,
CKWDS,
ETSG',
DFAD
PMUL,
PYLUP,
JLCLR,
'PLOOP,
'TEMP,
'577,
ORM,
TSIG,
BUMP,
DEPF,
1,
DIV,
VLP,
ZDIV,
FLOOP,
SN,
'43
FIX,    SUBROUTINE TO DO BASIC 'INT' TO FLOATING AC.(INT(-1.5)=-2);
```

```
        A POINTER TO IT.
LETTER, THERE WAS A LETTER WHICH MUST BE A VARIABLE,
        WHICH MAY OR MAY NOT BE FOLLOWED BY A DIGIT.
        MAKE THE VARIABLE NAME,
SIMPLV, AND SEARCH THE USER SYMBOL TABLE FOR THAT VARIABLE NAME.
VSCHLUP,IF IT IS FOUND, RETURN A POINTER TO IT.
ISDEF?,
ITSDEF,
SCHNOT,OTHERWISE PUT ONE IN,
        SET UP THE CORRESPONDING VARIABLE SPACE, AND RETURN A POINTER
        TO THAT,
.0002,  LITERAL,
TEN,    FLOATING POINT 10.0 .
POP,    PUSH DOWN LIST POPPER,
FOURLF, TEXT OF FOUR (CRLF)S.
UGH1,   LINE MAKES PROGRAM 'TOO BIG, LINE IGNORED' MESSAGE.
LIST,   DOES THE LIST COMMAND,
        CHECKS IF THERE IS A LINE NUMBER FOLLOWING THE COMMAND,
        IF THERE IS, AND IT IS DEFINED, LIST FROM THAT LINE,
LISTALL,OTHERWISE LIST FROM THE BEGINNING,
LISTSOM,
LISTLUP,LISTING IS STOPPED BY TYPING ANY CHARACTER,
        IF IT'S (EOF) THEN THE LISTING IS DONE,
        IF IT'S A LINE NUMBER, PRINT IT,
LIST2,  IF ITS A VARIABLE, PRINT IT,
PRINVAR,TEMPORARY WHICH HOLDS THE VARIABLE NAME,
LIST3,  IF IT'S A LITERAL, THEN PRINT IT,
LIST4,  IF IT'S TEXT, THEN PRINT IT
4LUP,   IN A LOOP,
LIST5,  OTHERWISE IT'S A SYSTEM SYMBOL, SO PRINT THAT,
PRINTXT,SUBROUTINE FOR PRINTING PACKED ASCII TEXT,
PRLOOP,
PRTXRET,
PRSUBP,
CRLFPR,
07741,  LITERAL,
PRINUM, ROUTINE FOR PRINT LINE NUMBERS,
        IT DOES IT BY CONVERTING TO FLOATING POINT(WHICH THE OUTPUT
        ROUTINE WILL PRINT IN INTEGER FORM),
0233,   LITERAL,
RESTORE,MUST BE FOLLOWED BY (CRLF) OR '\'
        RESETS THE READ-DATA POINTER,
BREAK,  'STOP'
READY,  'READY'
CRLF=   A CARRIAGE RETURN LINE FEED,
MOREIN, IF IT'S A (CRLF) OR '\' THEN ITS THE END OF THE 'INPUT' STATEMENT,
INPUT,  PRINT A '?',
        GET A LINE,(CHECKING FOR 'STOP') AND FOR EACH VARIABLE, USING
INPLUP, 'GETVAR' AND 'STOVAR', SWITCH THE LOCATION COUNTER FOR A POINTER
        INTO THE LINE BUFFER AND SCAN A VALUE(EXPRESSIONS OK,),
        AND SWITCH THE LOCATION COUNTER BACK AFTERWARD,
INPPTR, TEMPORARY,
INSTMP, TEMPORARY,
INLCTMP,TEMPORARY,
01765,  LITERAL,
PSTOP,  POINTER TO 'STOP' SYMBOL,
XGISIT, ROUTINE WHICH IMPLEMENTS 'GET=ISIT',
XISIT,  ROUTINE WHICH IMPLEMENTS 'ISIT',
ISITLIT,ROUTINE WHICH CHECKS FOR AND ACCUMULATES LITERALS(AT RUNTIME),
        2000+N FROM @WORD MEANS AN N WORD LITERAL,
```

```
        .1.1.1.1.1.1.1.1.1.1.1.1.1.1 ETC. CHALK IT UP TO 'LINE TOO L(NG'.
PTUBIG, POINTER TO 'LINE TOO LONG',
PGETLBE,POINTER TO RETURN FROM 'GETLIN',
NCTCR,  ITS NOT A CARRIAGE RETURN SO
        IF ITS A DIGIT OR '.', GO ACCUMULATE A LITERAL,
        IF ITS A LETTER, GO LOOKUP OR CREATE A VARIABLE.
REMPACK,OTHERWISE IT IS QUOTED(OR UNQUOTED) TEXT.
TXTPAK, PACK TWO SIXBIT CHARACTERS PER WORD.
LHALF,  END WITH ONE OR TWO ZERO CHARACTERS(AT SIGNS DONT EXIST),
RHALF,  AND
CRINTXT,CARRIAGE RETURN FORCES END OF TEXT,
DQINTXT,SO DOES ANOTHER DOUBLE QUOTE,
MTXXIT, THEN TRY FOR ANOTHER SYMBOL,
PLETTER,WHERE TO GO TO MAKE OR FIND A VARIABLE,
07753,  LITERALS
042??,
042,
THISTXT,TEMPORARY,
NONBLNK,ROUTINE WHICH SKIPS OVER BLANKS(EVERYWHERE EXCEPT TEXT),
032,    LITERALS,
07737,
PXXCRLF,POINTER TO (CRLF) SYMBOL,
PXXEXIT,POINTER TO (EXIT) SYMBOL,
LITRAL, CREATE A LITERAL OUT OF THE FLOATING POINT AC,
        DEPENDING ON HOW MANY LOW ORDER ZEROES, IT CAN BE
JUST1,  12 BITS,
ALL3,   36 BITS,
JUST2,  24 BITS, OR
JUST?,  0 BITS LONG,
POPER?, AUTO INDEX POINTER TO FLOATING POINT TEMPORARY,
PXXLIT0,POINTER TO ZERO LENGTH LITERAL SYMBOL,
DIGIT,  A NUMBER IS TO BE ACCUMULATED, SO SET 'SNUMFLG' AS A FLAG
        TO TELL WHETHER IT SHOULD BE A LITERAL OR A STATEMENT NUMBER,
DIGIN,  SCAN DIGIT BY DIGIT MULTIPLYING BY 10.0,
ITSDP,  AND COUNTING THE NUMBER OF DIGITS AFTER A DECIMAL POINT,
ITSE,   AND WATCHING OUT FOR 'E' EXPONENT,
ITSP,   ACCUMULATE THE EXPONENT
NOTSGN, EITHER PLUS OR MINUS,
07673,  LITERAL,(WHICH IS NOT EXECUTED SINCE THE PREV, INSTRUCTION
        MUST SKIP),
ONLY1,
ENDNUM, AND SUBTRACT THE NUMBER OF DIGITS AFTER THE DECIMAL POINT,
        THEN SCALE THE NUMBER UP OR DOWN,
MULEXP, MULTIPLY BY 10.0 OR DIVIDE BY 10.0,
STATNO, NOW COMES THAT FLAG FOR LINE NUMBER OR LITERAL,
STATNO, MAKE A LINE NUMBER OUT OF IT,
FDIGIT, FLOATING POINT TEMPORARY FOR ADDING DIGITS ARE THEY ARE HIT,
MULXTEN,MULTIPLY BY 10,0 INSTRUCTION,
DIVXTEN,WHEN ADDED TO 'MULXTEN' MAKES A DIVIDE BY 10.0 INSTRUCTION,
DPFLAG, DECIMAL POINT FLAG,
DECFRAC,NUMBER OF DIGITS AFTER THE DECIMAL POINT,
ISDIG,  SPACE SAVER WHICH CHECKS IF THE CHARACTER IS A DIGIT,
027,    LITERALS,
01742,
01774,
PLITRAL,POINTER TO LITERAL MAKER,
COMMON, SEARCH THROUGH
LUP,    USER SYMBOL TABLE FOR A GIVEN LINE NUMBER
NOT,    AND IF IT IS IN THERE RETURN A POINTER TO IT
IN,     OTHERWISE PUT IT IN THERE AND THEN RETURN
```

```
RUN2LOP, AND GO THROUGH THE USER SYMBOL TABLE AGAIN, REALLOCATING
RUN2NOT, SPACE FOR A SCALAR FOR EACH VARIABLE.
UN2IN,
        ALSO SET UP THE LOCATION COUNTER, PUSH DOWN LIST, READ POINTER,
        FOR TABLE, AND GOSUB TABLE FOR EXECUTION OF A 'RUN'.
LIMIT,  POINTER TO FIRST WORD OF USER SPACE(VARIABLE SPACE STARTS HERE).
XXTHEN, POINTER TO 'THEN' SYMBOL(FAKES LINE GETTER INTO STARTING
        THE LINE WITH A LINE NUMBER).
IST,    POINTER TO 'LIS' PROCESSOR,
XXEOF,  POINTER TO (EOF) SYMBOL.
ERMSY,  POINTER TO PERM, SYMBOL TABLE(USER SYMBOL TABLE IS BELOW IT).
OTNOW,  POINTER TO INDIRECT STATEMENT HANDLER(INSERTS LINES).
ND,     THE END STATEMENT IS EXECUTEABLE, IT IS NOT EQUIVALENT TO STOP.
        IT CLEARS VARIABLES, THE REASON FOR THIS IS THAT AFTER A STOP,
        ALL ARRAYS WHICH WERE DEFINED AT THE TIME OF THE STOP REMAIN
        DEFINED(INTERACTIVE!). IF THE USER THEN CONTINUES TYPING MORE
        PROGRAM, HE WILL EVENTUALLY COME UP AGAINST THE 'TOO-BIG ERROR'.
        HE HAS TO HAVE A WAY OF CLEARING HIS ARRAYS TO GET PAST THE
        ERROR, AND RATHER THAN ADDING A COMMAND, THE END STATEMENT WAS
        UTILIZED. SO THE WAY TO GET PAST THE 'TOO-BIG ERROR' IN THIS CASE IS
        TO TYPE A DIRECT 'END' STATEMENT.
STOP,   PRINT 'READY' AND GO TO WAIT FOR A TELETYPE COMMAND.
TOCLONG, THE 'LINE TOO LONG' MESSAGE.
DELETED, THE 'DELETED' MESSAGE.
GETLRET, FOR RETURNING FROM 'GETLIN' FROM OFF PAGE,
        AFTER INSERTING A LINE, FIX UP LINENO DEFINITIONS,
        RETURN.
ETLIN,  ROUTINE FOR GETTING A TRANSLATED(INTO INTERPRETIVE CODE FORM)
        LINE FROM THE TELETYPE. IT IS ENTERED WITH SOMETHING
        IN THE AC-THAT SOMETHING BEING A POINTER TO 'THEN' IF THE LINE
        IS MEANT TO START WITH A LINE NUMBER(AS OPPOSED TO A LITERAL).
NEWLI,  GET A NEW LINE,
NEWCHAR, ONE CHARACTER PER WORD IN TRIMMED ASCII
        WATCHING OUT FOR 175-ALT, 176-ALT, '←' '@' AND (CR),
        IGNORE NONPRINTING CHARACTERS(AND '@').
TUBIG,  IF THE LINE IS BIGGER THAN THE BUFFER CAN HANDLE, PRINT
        'LINE TOO LONG' AND TRY FOR A NEW LINE,
ALTMODE, PRINT 'DELETED' AND TRY FOR A NEW LINE,
BARROW, REMOVE PREVIOUS CHARACTER(IF THERE WAS ONE),
CARRET, NOW THERE IS A WHOLE LINE OF ASCII IN THE BUFFER, START
SLOOP,  SCANNING OFF SYMBOLS ONE AT A TIME, FROM LEFT TO RIGHT,

GLOOP,  IF THERE IS A MATCH WITH THE PERMANENT SYMBOL TABLE, USE THAT
        AS THE DEFINITION(IGNORE ALL BLANKS IN THE PERM, SYMBOL TABLE).
HLOOP,
BSKIP,
AMATCH, WATCH OUT FOR THE SPECIAL CASE OF 'REM'(STUFF AFTER 'REM' IS
        TREATED AS IF IT WAS QUOTED TEXT).
036,    LITERALS,
040,
0122,
07622,
ROTHER, POINTER TO WHERE TO GO IF THERE IS NO MATCH IN THE PERM, SYMBOL
        TABLE.
PTABLE, POINTER TO THE PERMANENT SYMBOL TABLE.
OTHER,  THERE WAS NO MATCH WITH THE PERMANENT SYMBOL TABLE.
        IF THE BEGINNING CHARACTER IS A CARRIAGE RETURN, THEN FILL OUT
        THE LINE WITH (EXIT)S AND CHECK FOR A REAL HAIRY LINE WHICH
        CAN OCCUR BECAUSE OF THE FACT THAT LITERALS CAN BE LARGER
        AFTER TRANSLATION THAN BEFORE, IF IT REALLY WAS A LINE LIKE
```

```
        THE INTERPRETIVE CODE IS STORED WITHOUT HOLES IN IT.
NOTFST,
MOVE.,
TPLACE,
TRALLP,
FIXLIN, AFTER THIS INSERTION, THE LINE NUMBER DEFINITIONS MUST
        BE FIXED, SO DO IT.
PASSOP, ROUTINE TO LOOK THROUGH INTERPRETIVE CODE FOR A CARRIAGE
        RETURN SYMBOL WHEN LOOKING AT A LINE AT A TIME.
NOTEAD, SPECIAL CASES TO BE HANDLED ARE QUOTED TEXT AND LITERALS.
COMPARE,COMPARE THE NUMERIC LINE NUMBER WITH THE NUMBER IN
        OPERAND,OPERAND+1 AND SKIP IF OPERAND,OPERAND+1 IS LESS.
LSTLOC, TEMPORARY.
SUBRA,  ROUTINE USED IN INSERTION DELETION FOR SIZE OPTIMIZATION.
LOWLOC, TEMPORARY.
PUTLOC, TEMPORARY.
PRINT,  IF NEXT SYMBOL IS (CRLF) OR '\'
PPRINCR, PRINT A CARRIAGE RETURN LINE FEED.
NOPCR,  IF ITS A ';' FOLLOWED BY (CRLF) OR '\',JUST RETURN.
        IF ITS A ',' THEN SPACE OUT TO FILL FIELD.
PRINTHS,IF A NEW ELEMENT IS TO BE PRINTED AND THE COLUMN POINTER
        IS PAST 58, THEN INSERT A CARRIAGE RETURN LINE FEED.
PRINQUO,IF THE SYMBOL IS QUOTED TEXT PRINT IT AS IS.
NOTTXT, OTHERWISE EVALUATE IT AND PRINT AS A NUMBER.
PEVALGO,POINTER FOR FAKING THE JMS TO EVAL.
PPRINRE,POINTER FOR FAKING THE JMS TO EVAL.
PRINRET,THIS PRINTS IT AS A NUMBER WITH A SPACE BEFORE IT
        IF IT IS POSITIVE AND A SPACE AFTER, REGARDLESS.
PRINHAF,ROUTINE FOR PRINTING A HALFWORD OF THE QUOTED TEXT.
PRINBLK,PRINT ONE BLANK OF THE ',' FIELD FILL.
PRINCOM,PRINT BLANKS UNTIL COLUMS 14,28,42, OR 56.
PRINSEM,TREAT SEMICOLONS IN PRINT.
007736, LITERALS.
07762.
LINFIX, ROUTINE WHICH DOES THE ACTUAL FIXING UP OF LINE NUMBER
        DEFINITIONS AFTER THE INTERPRETIVE CODE HAS BEEN SHIFTED AROUND.
LFXLOP,
PPASSCR,POINTER TO THE ROUTINE WHICH SKIPS TILL CARRIAGE RETURN.
PUSH,   PUSH DOWN LIST PUSHER.
MENDPDL,ONES COMPLEMENT POINTER TO END OF PUSH DOWN LIST.
MOREDIM,THERE IS MORE TO BE DIMENSIONED. MUST BE A ',' HERE.
DIM,    SET 'DIMFLAG' AND CALL THE EVALUATOR. IT WILL DO THE ALLOCATION.
EXIT,   AFTER A DIRECT STATEMENT COME HERE.
        PRINT CARRIAGE RETURN,
EDIT,   AND LINE FEED
        AND GET A NEW LINE,
        IF IT STARTS WITH A LINE NUMBER, IT IS AN INDIRECT COMMAND
        SO INSERT IT.
        IF IT STARTS WITH 'RUN','LIS', OR 'SCR' DO THEM.
        OTHERWISE IT IS AN IMMEDIATE MODE STATEMENT.
SCRATCH,MAKE THE USER SYMBOL TABLE HAVE ZERO ENTRIES, AND THE
        INTERPRETIVE CODE BE JUST AN (EOF).
RUN,    CLEAR ALL VARIABLES TO SCALARS WITH VALUE 0.0.
        AND START EXECUTION OF INTERPRETIVE CODE.
IMMED,  START EXECUTION FROM INPUT LINE BUFFER.
CLEARV, ROUTINE WHICH DE-DEFINES ARRAYS AND CLEARS VARIABLES TO 0.0.
RUNLOP, GOING THROUGH THE USER SYMBOL TABLE, RETRIEVE THE VARIABLE
RUNNOT, NAMES(FROM VARIABLE SPACE) AND SAVE THEM IN THE SYMBOL TABLE
RUNIN,  ENTRY IN PLACE OF THE POINTER.
        THEN CLEAR THE VARIABLE SPACE.
```

```
          OF EXPRESSION.
023,      LITERAL.
CONST,    TABLE FOR SELECTING THE PROPER RELATIONAL SKIP.
GETADDR,  ROUTINE WHICH TAKES THREE ARGUMENTS.
          POINTER TO THE SYMBOL TABLE ENTRY FOR A VARIABLE.
          VALUE OF SUBSCRIPT ONE.
          VALUE OF SUBSCRIPT TWO.
          AND STORES A POINTER INTO THE VARIABLE SPACE CORRESPONDING
          TO THE VALUE OF THIS VARIABLE.
NOSS2,
NOSS1,
ALLOC,    IF THE REFERENCE CALLS FOR SUBSCRIPTS BUT THE VARIABLE
          IS UNSUBSCRIPTED, IT GETS DIMENSIONED (0:10) OR (0:10,0:10).
ISDIM,    UNLESS THIS IS A DIMENSION STATEMENT IN WHICH CASE THE ACTUAL
          VALUES OF THE SUBSCRIPTS ARE USED AS DIMENSIONING INFORMATION.
GSS1,     TEMPORARIES USED IN GETADDR.
GSS2,
GDIM2,
PSPACER,  POINTER TO 'TOO-BIG ERROR'
07774,    LITERALS.
013,      THIS ONE ISNT QUITE A LITERAL, ITS EQUAL TO 12 OCTAL IF THE
          USER ANSWERS 1 TO THE 'DO SUBSCRIPTS START AT 0 OR 1'.
SSERR,    PRINT 'SUBSCRIPT ERROR'
DEF,      SCANS A 'DEF' STATEMENT AND SAVES THE ADDRESS
          OF THE LINE IN 'USERFN', IT IGNORES THE WHOLE LINE.
PSKIPIT,  POINTER TO PLACE WHICH SKIPS THE LINE.
USERFN,   HOLDS THE ADDRESS OF THE ACTIVE DEF.
MORERD,   PART OF THE READ. THERE MUST BE A ',' HERE
READ,     GET THE VARIABLE TO STORE INTO.
          SWITCH TO READ-DATA LOCATION COUNTER.
          AND LOOK FOR ','.
ISSOME,   THERE IS SOME DATA SO EVALUATE IT(EXPRESSIONS ARE OK).
          SWITCH BACK TO REAL LOCATION COUNTER.
          AND STORE THE VARIABLE.
SCHNOP,
SEARCH,   SEARCH FOR SOME 'DATA' STATEMENTS.
          IF (EOF) IS ENCOUNTERED BEFORE 'DATA' THEN
DATAERR,  PRINT 'DATA ERROR'.
LOCTMP,   TEMPORARY.
GETBLK,   ALLOCATES A 4 WORD BLOCK FOR SYMBOL TABLE.
GTBKLP,   IT HAS TO MOVE THE PROGRAM (CODE) DOWN 4 WORDS IN ORDER TO
          DO IT. THIS MEANS RELOCATION OF THE FOR TABLE, AND THE
          GOSUB TABLE ENTRIES.
ABCDEF,   LITERAL.
BCDEFG,   TEMPORARY.
PPFORLI,  POINTER
TTY,      A USELESS FUNCTION 'TTY' WHICH TYPES AN ASCII CHARACTER AS
          A SIDE EFFECT. THIS MAY BE OVERLAID AND THE NAME CHANGED IN
          THE PERMANENT SYMBOL TABLE, TO ANY OTHER THREE CHARACTER NAME.
          NOTE THAT IT IS JUST A SIMPLE JMS'D TO SUBROUTINE WHICH
          HAS ITS ARGUMENT IN ACS,ACE,AC1,AC2,AC3 AND RETURNS ITS VALUE
          THERE ALSO.
NOTNOW,   A COMMAND LINE THAT STARTS WITH A LINE NUMBER IS NOT TO BE
          EXECUTED NOW.
          IF THE LINE NUMBER IS DEFINED,
MOVE,     THE OLD LINE MUST BE REMOVED FROM THE INTERPRETIVE CODE BEFORE
          THE NEW ONE MAY BE INSERTED.
INSERTS,
INSERT,   INSERT THE LINE IN 'LINBUF' INTO THE INTERPRETIVE CODE.
          'FPTR' IS THE LAST PLACE THAT WAS STORED IN TO CREATE IT.
```

```
SGN,     ROUTINE WHICH DOES BASIC 'SGN',
MNSC..,  FLOATING POINT -1.0,
GETCH,   GET CHARACTER FROM TTY ROUTINE,
PUTCH,   PUT CHARACTER TO TTY ROUTINE,
         KEEPS TRACK OF COLUMN POINTER SO PRINT KNOWS WHERE IT IS,
START,   MANUAL STARTING ADDRESS. LEAVES MACHINE AS IF 'SCR' WAS JUST
         TYPED,
EVAL,    EXPRESSION EVALUATORE ROUTINE, IS JMS'D TO INSIDE ITSELF,
EVALGO,  SOMETHING FAKES A JMS TO EVAL AND ENTERS HERE,
         PUSH A BEGINNING TERMINATOR ON THE STACK(SAME PRECEDENCE AS
         END OF EXPRESSION),
ISUMIN,  IT IS A UNARY MINUS,
         SO PUSH THE SPECIAL CODE FOR UNARY MINUS ON THE STACK,
GETOPR,  GET AN OPERAND(WHICH INCLUDES A UNARY MINUS AND A UNARY PLUS),
         '(' EXPRESSION ')' IS ALSO AN OPERAND EVALUATED RECURSIVELY,
NOPAREN, A VARIABLE IS AN OPERAND,
         WHICH CAN BE SUBSCRIPTED(MORE RECURSION),
ONEDIM,  BY ONE OR TWO SUBSCRIPTS,
GOTSS,   GET THE ADDRESS OF THE VARIABLE SPACE,
WDTEMP,  FROM THE ADDRESS OF THE VARIABLE SYMBOL,
ONESS,   AND THE FIRST SUBSCRIPT VALUE,
TWOSS,   AND THE SECOND SUBSCRIPT VALUE,
NOTVAR,  A LITERAL IS AN OPERAND,
ISITFUN, AND A FUNCTION IS AN OPERAND,
         ANYTHING ELSE GIVES A 'SYNTAX ERROR',
         'FN' IS A SPECIAL CASE FUNCTION WHICH MUST SKIP OVER A VARIABLE
         'FNA' IS 'FN'(A KEYWORD) CAT 'A'(A VARIABLE),
         GET THE ARGUMENT, AND
JUMP,    JMS THROUGH
FUNTAB,  THIS JUMP TABLE,
04014,   LITERALS
04213,
XGMUST,  THE ROUTINE WHICH IMPLEMENTS 'GET*MUSTBE',
XMUST,   THE ROUTINE WHICH IMPLEMENTS 'MUSTBE',
SXERR,   PRINT 'SYNTAX ERROR',
ATLINE,  TEXT FOR MESSAGES LIKE 'SYNTAX ERROR AT LINE 250',
FNEXIT,  WHEN A FUNCTION RETURNS, THERE MUST BE A ')' AS THE CURRENT
         SYMBOL,
GOTOPR,  HAVING JUST GOTTEN AN OPERAND
..DONE,  AN OPERATOR(OR END OF LINE) MUST NOW BE GOTTEN,
         THE PRECEDENCE IS IN THE 0700 POSITION OF THE SYMBOL IDENTIFIER
         NUMBER(REMEMBER THE EXPLANATION OF 'WORD'),
         IF THE PRECEDENCE OF THIS OPERATOR IS  GREATER THAN THE PREVIOUS
         THEN STACK EVERYTHING AND GET ANOTHER OPERAND,
DOITNOW, OTHERWISE DO THE PREVIOUS OPERATOR NOW,
OTEMP,   JUMP TO THE CORRECT ROUTINE,
0 377,   LITERAL,
OJUMP,   THIS IS THE OPERATOR JUMP TABLE,
PLUS,    FLOATING ADD,
FADEXT,
MINUS,   FLOATING SUBTRACT,
UNOPER,  FLOATING NEGATE,
STAR,    FLOATING MULTIPLY,
SLASH,   FLOATING DIVIDE,
SLSHTMP, FLOATING POINT TEMPORARY FOR DIVIDE(INVERSE DIVIDE),
07570,   LITERAL,
UPARROW, FLOATING POINT POWER(NO EXTENDED FUNCTIONS),
RELATE,  RELATIONALS(RETURN 1.0=.TRUE., 0.0=.FALSE.),
THESKIP, THE CORRECT FLOATING POINT SKIP FOR THE RELATIONAL,
ENDOPER, EXECUTING THE BEGINNING TERMINATOR AS AN OPERATOR MEANS END
```

```
        THAT LINE NUMBER. IF ,TRUE, AND NOT A LINE
        NUMBER, EXECUTE IT AS THE BEGINNING OF A STATEMENT.
XEXECU, POINTER,
FOR,    NEXT SYMBOL MUST BE A VARIABLE,
        IF THE VARIABLE IS ALREADYIN THE FOR TABLE,
LUPF,
INLUPF, THEN COMPRESS IT OUT.
NOTHERE,IF THERE ARE ALREADY 8 VARIABLES IN THE FOR TABLE,
        THEN GIVE THE 'FOR ERROR' SINCE 8 IS THE LIMIT,
        NEXT SYMBOL MUST BE AN '=',
        FOLLOWED BY AN EXPRESSION,
        SET THE FOR VARIABLE TO THIS EXPRESSION,
        CHECK THAT THE NEXT SYMBOL IS 'TO',
        THEN PUT THE VARIABLE IN THE FOR TABLE ALONG WITH A POINTER
        TO HERE,
SKIPIT, SKIP THE CODE UP TO A (CRLF) OR '\',
FORERR, PRINT 'FOR ERROR',
011,    LITERAL,
GOSUB,  NEXT TWO SYMBOLS MUST BE LINE NUMBER AND (CRLF) OR '\',
        CHECK THAT THERE IS ROOM IN THE GOSUB TABLE,
        IF NOT, PRINT 'GOSUB ERROR', OTHERWISE PUT THE RETURN POINTER
        IN THE GOSUB TABLE, AND DO A GOTO IT,
GOTO,   NEXT TWO SYMBOLS MUST BE LINE NUMBER AND (CRLF) OR '\',
        THE NEW LOCATION COUNTER IS GOTTEN FROM THE LINE NUMBER,
MSGERND,TWOS COMPLEMENT POINTER
DEEPERR,PRINT 'GOSUB ERROR'
GOBOTH, ROUTINE USED BY GOSUB AND GOTO TO GET A LINE NUMBER,
        SEE IF IT IS DEFINED,
GOERR,  PRINT 'LINENO ERROR' IF ITS NOT DEFINED,
ISITOF, AND MAKE SURE THAT THE NEXT SYMBOL IS (CRLF) OR '\',
VARTEMP,FLOATING POINT TEMPORARY USED DURING FOR-NEXT,
PXFORLI,POINTER TO FOR TABLE FOR AUTO INDEX,
FINDIT, ROUTINE WHICH SEARCHES THE FOR TABLE AND SKIPS WHEN AND IF
        IT FINDS WHAT IT IS LOOKING FOR,
FINDLUP,INDEX1 IS LEFT SET UP,
FOUND,  AND SO IS GOTEMP(AS THE COUNTER),
        RETURN,
STICKIT,ROUTINE USED TO ZERO VARIABLES,(MAINLY A SPACE SAVER),
NEXT,   NEXT SYMBOL MUST BE A VARIABLE,
        WHICH IS IN THE FOR TABLE, ELSE PRINT 'NEXT ERROR',
        FOLLOWED BY (CRLF) OR '\',
        EVALUATE FROM POINTER IN FOR TABLE,
TRYSTEP,CHECK FOR 'STEP' AND IF SO EVALUATE THE STEP,
        MUST BE FOLLOWED BY (CRLF) OR '\',
GOTSTEP,STEP IS IMPLIED ONE,
        ADD THE STEP TO THE VARIABLE,AND CHECK TO SEE IF THE LOOP
        IS ENDED, IF NOT JUST CONTINUE EXECUTING FROM HERE,
FORDONE,OTHERWISE RESET THE LOCATION COUNTER TO AFTER THE NEXT,
LOCTEMP,TEMPORARY,
PFINDIT,POINTER TO FOR TABLE LOOKER UPPER,
NEXTERR,PRINT 'NEXT ERROR',
RETURN, NEXT SYMBOL MUST BE (CRLF) OR '\',
        CHECK FOR RETURN BEFORE GOSUB, IF ITS NOT, THEN SET THE
        LOCATION COUNTER FROM IT BY POPPING,
RETNERR,OTHERWISE PRINT 'RETURN ERROR',
MGOLIST,NEGATIVE POINTER,
FORLIM, FLOATING POINT TEMPORARY FOR FOR LOOP LIMIT,
FORSTEP,FLOATING POINT TEMPORARY FOR FOR LOOP STEP(MUST DIRECTLY PRECEED
        A SCRATCH LOCATION WITH RESPECT TO NEXT),
INT,    ROUTINE WHICH DOES BASIC 'INT',
```

```
02000,  LITERAL,
FPFLAG, FLAG USED BY FLOATING POINT PACK TO CHECK FOR ZERO OPERANDS.
POADD,  POINTER TO ROUTINE WHICH ADDS (AC1,AC2,AC3)+(OP1,OP2,OP3).
04,     LITERAL,
PLINFIX,POINTER TO ROUTINE WHICH FIXES STATEMENT NUMBER DEFINITIONS
        AFTER A LINE HAS BEEN INSERTED.
07745,  LITERAL,
PCHKFIT,POINTER TO ROUTINE WHICH CHECKS TO SEE IF A LINE OR BLOCK FITS.
03755,  LITERAL,
PGOLIST,POINTER TO THE GOSUB/RETURN LIST,
GSBPTR, POINTER TO CURRENT GOSUB RETURN.
GET=    USED WITH 'ISIT' AND 'MUSTBE' FOR GETTING NEXT PROGRAM ELEMENT.
ISIT=   SUBROUTINE INVOCATION TO SKIP IF A CERTAIN PROGRAM ELEMENT IS
        CURRENTLY BEING LOOKED AT.
MUSTBE= SUBROUTINE INVOCATION TO GIVE A 'SYNTAX ERROR' IF A CERTAIN
        PROGRAM ELEMENT IS NOT BEING LOOKED AT.
PLBEGIN,POINTER TO SECTION OF LINE INPUT BUFFER USED BEFORE TRANSLATION.
MLBEGIN,ONES COMPLEMENT OF PLBEGIN,
MLEND,  TWOS COMPLEMENT OF POINTER TO END OF PUSH DOWN LIST,(ALSO USED
        AS PART OF LINE INPUT BUFFER),
PISITLI,POINTER TO ROUTINE WHICH CHECKS IF A LITERAL AND ACCUMULATES IT.
GETWD,  ROUTINE THAT SAVES CURRENT ELEMENT IN 'WORD' AND BUMPS 'LOCCTR'.
PLETDO, POINTER USED TO FAKE A JMS TO PART OF 'GETVAR',
LETDO,  MAKE SURE THAT CURRENT SYMBOL IS AN '=',
        CALL EXPRESSION EVALUATOR,
        AND MAKE SURE THAT NEXT SYMBOL IS (CRLF) OR '\',
        THEN STORE THE VALUE IN THE VARIABLE WHICH HAS BEEN SET UP.
EXECUTE,IF THERE IS AN 'S' IN THE KEYBOARD READER BUFFER, THEN
        PRINT STOP-READY AND GO TO 'EDIT',
07655,  LITERAL,
NOBREAK,SET UP 'GETVAR' SO THAT IT CAN BE JUMPED INTO,
        IF THE CURRENT SYMBOL IS NOT A KEYWORD, ASSUME ITS A 'LET' STMT.
        IF THE CURRENT SYMBOL IS A LINE NUMBER, SAVE IT IN 'LINENO',
KEYWD,  IF THE CURRENT SYMBOL IS A STATEMENT WORD, BRANCH ON IT.
SJUMP,  (USING THIS JUMP TABLE),
GETVAR, ROUTINE WHICH SCANS A VARIABLE AND SETS UP 'STOVAR' TO STORE
        INTO IT,
LET,    LOOK AT NEXT SYMBOL,
NOTCAD, WHICH MUST BE A VARIABLE,
        IF THERE ARE SUBSCRIPTS, GET THEM,
NOCOMMA,UNLESS THAS WERE ONLY ONE, IN WHICH CASE COME HERE,
        AND RETURN,
STOVAR, ROUTINE WHICH STORES IN A VARIABLE SET UP BY 'GETVAR',
VAR,    POINTER TO THE VARIABLE,
SSONE,  VALUE OF THE FIRST SUBSCRIPT+1(ZERO FOR SCALARS),
SSTWO,  VALUE OF SECOND SUBSCRIPT,
BADERR, THIS IS WHERE UNDEFINED FUNCTIONS POINT,
FNERR,  PRINT 'FUNCTION ERROR',
SPACERR,PRINT 'TOO-BIG ERROR',
CHKFIT, ROUTINE TO CHECK THAT A LINE OR A SYMBOL TABLE BLOCK WILL NOT
        MAKE PROGRAM TOO BIG TO FIT,
        'ARRLOC' MUST CONTAIN LESS THAN 'CODELOC', IF THEY WOULD
        CROSS, THEN IS DOESNT FIT,
        IF IT DOESN'T FIT, PRINT 'TOO BIG, LINE IGNORED' AND TREAT
        AS ALT-MODE,
ABS,    ROUTINE WHICH DOES BASIC 'ABS',
IF,     CALL EXPRESSION EVALUATOR,
        NEXT SYMBOL MUST BE 'THEN'(WHICH IS A PSEUDONYM FOR 'GOTO'),
        GREATER THAN ZERO MEANS .TRUE., LESS THAN OR EQUAL MEANS .FALSE.
        IF .TRUE. AND THE NEXT SYMBOL IS A LINE NUMBER, THEN DO A 'GOTO'
```

```
GPTR,   DURING THE LINE TRANSLATION PROCESS, THIS POINTS TO THE CURRENT
        CHARACTER.
HPTR,   DURING THE LINE TRANSLATION PROCESS, THIS POINTS TO THE CURRENT
        PERMANENT SYMBOL TABLE ENTRY BEING MATCHED AGAINST.
02,     LITERAL.
FORCT,  ONES COMPLEMENT OF THE NUMBER OF FOR LOOPS WHICH HAVE BEEN
        STARTED BUT NOT FINISHED.
SNUMFLG,DURING THE LINE TRANSLATION PROCESS, THIS IS USED TO TELL
        WHETHER OR NOT THE LAST SCANNED SYMBOL WAS XXCRLF,XXTHEN,
        XX3OSUB, OR XXGOTO.
07772,  LITERALS.
012,    12
03737,
OLDOP,  DURING EXPRESSION EVALUATION, THIS IS THE PREVIOUS OPERATOR.
ADDRESS,THIS IS USED AS AN INDIRECT POINTER FOR FETCHING AND STORING
        BASIC VARIABLES.
0700,   LITERALS.
077,
 7764,
(377,
(7,
OPERAND,DURING EXPRESSION EVALUATION, THIS IS A FLOATING POINT TEMPORARY
        WHICH IS THE OPERAND OF THE CURRENT OPERATION.
PTEXT,  POINTER TO 'XXTEXT' THE INTERPRETERS QUOTED TEXT INDICATOR.
PERROR, POINTER TO THE ERROR MESSAGE ROUTINE.
PSXERR, POINTER TO 'SYNTAX ERROR'
PEVAL,  POINTER TO EXPRESSION EVALUATOR.
PGETADD,POINTER TO THE ADDRESS GETTER OF BASIC VARIABLES.
XECUT,  POINTER TO THE INTERPRETER MAIN LOOP.
PPUSH,  POINTER TO PUSH DOWN LIST PUSHER.
PPOP,   POINTER TO PUSH DOWN LIST POPPER.
PFIX,   POINTER TO ROUTINE WHICH UNNORMALIZES THE FLOATING AC, IT IS ONLY
        USEFUL FOR SUBSCRIPTING SINCE IT RETURNS INT(AC)-INDEX ORIGIN.
PGOTOPR,POINTER INTO EVALUATOR WHERE THE OPERAND WAS JUST GOTTEN.
PGETOPR,POINTER INTO EVALUATOR WHERE THE OPERAND MUST BE GOTTEN.
PSTOVAR,POINTER TO ROUTINE WHICH STORES THE VARIABLE SET UP BY 'GETVAR'.
PGETVAR,POINTER TO ROUTINE USED FOR ASSIGNMENT, READ, AND INPUT.
PRINTX,POINTER TO ROUTINE FOR PRINTING PACKED ASCII.
GETBLK,POINTER TO ROUTINE WHICH ALLOCATES FOR USER SYMBOL TABLE.
SLOOP,  POINTER TO MAIN LOOP IN LINE TRANSLATION PROCESS.
OUTNUM,POINTER TO ROUTINE WHICH PRINTS A FLOATING POINT NUMBER.
EDIT,   POINTER TO SYSTEM LOOP WHERE BASIC WAITS FOR A COMMAND.
2047,   LITERAL.
FNERR,  POINTER TO 'FUNCTION ERROR'.
STICKI,POINTER TO ROUTINE USED FOR SETTING UP VARIABLE SPACE.
NONBLN,POINTER TO ROUTINEWHICH IGNORES BLANKS DURING LINE TRANSLATION.
PRINUM,POINTER TO ROUTINE WHICH PRINTS A LINE NUMBER.
GETLIN,POINTER TO ROUTINE WHICH GETS A TRANSLATED LINE FROM THE TTY.
7775,   LITERAL.
PDLIST,POINTER TO BEGINNING OF PUSH DOWN LIST.
COLUM,  POINTER TO CURRENT COLUMN WHICH THE TELETYPE IS PRINTING IN.
 5,     LITERALS.
 14,
 144,
 12,    POINTER TO FLOATING POINT CONSTANT 10.0.
ANORM,  POINTER FOR FLOATING POINT NORMALIZE ROUTINE.
AR1,    POINTER TO ROUTINE WHICH RIGHT SHIFTS AC1,AC2,AC3.
AL1,    POINTER TO ROUTINE WHICH LEFT SHIFTS AC1,AC2,AC3.
NEG,    FLOATING POINT 1.2(MUST DIRECTLY PRECEDE 'ZERO').
ZERO,   FLOATING POINT 0.2.
```

```
077..,  LITERALS(CONSTANTS WHICH NEVER CHANGE).
0215,
1212,
0772?,
0770?,
07763,
0260,
07771,
0177,
07000,
INDEX1, AN AUTO INDEX, ESSENTIALLY SCRATCH,
INDEX2, AN AUTO INDEX, ESSENTIALLY SCRATCH,
AC3,    LOW ORDER MANTISSA OF THE FLOATING AC,
        (AC3,AC2,AC1,OP3,OP2,OP1 ARE REFERENCED AS A BLOCK!!)
AC2,    MID ORDER MANTISSA OF THE FLOATING AC,
AC1,    HIGH ORDER MANTISSA OF THE FLOATING AC,
OP3,    LOW ORDER MANTISSA OF THE OPERAND IN FLOATING POINT PACK,
OP2,    MID ORDER MANTISSA OF THE OPERAND IN FLOATING POINT PACK,
OP1,    HIGH ORDER MANTISSA OF THE OPERAND IN FLOATING POINT PACK,
ACS,    SIGN OF THE FLOATING AC,
ACE,    EXPONENT OF THE FLOATING AC,
OPS,    SIGN OF THE OPERAND IN FLOATING POINT PACK,
OPE,    EXPONENT OF THE OPERAND IN FLOATING POINT PACK,
OV,     OVERFLOW WORD IN THE FLOATING POINT PACK(FOR ROUNDING),
TMP,    TEMPORARY USED BY THE FLOATING POINT PACK,
PGETCH, POINTER TO THE GET CHARACTER ROUTINE,
PPUTCH, POINTER TO THE PUT CHARACTER ROUTINE,
DIMFLAG,FLAG USED WHILE SUBSCRIPTING TO TELL WHETHER IT'S A DIM STMT,
FENTER= DEFINITION OF THE FLOATING POINT INTERPRETER ENTRY,
PDL,    ACTIVE POINTER FOR FILLING AND EMPTYING THE PUSH DOWN LIST,
PLINBUF,POINTER TO THE LINE INPUT BUFFER,
MLINBUF,TWOS COMPLEMENT OF POINTER TO THE LINE INPUT BUFFER,
MENDLIN,TWOS COMPLEMENT OF POINTER TO END OF LINE INPUT BUFFER,
PRTEMP, TEMPORARY USED BY TEXT PRINTING SUBROUTINE,
DECEXP, TEMPORARY DECIMAL EXPONENT USED IN I/O CONVERSION,
0256,   LITERAL,
PNUMBUF,POINTER TO NUMBER BUFFER(PART OF OUTPUT CONVERSION),
LOCCTR, LOCATION COUNTER OF INTERPRETER,
READLOC,LOCATION COUNTER FOR READ/DATA SCANNING,
PARGERR,POINTER TO 'ARGUMENT ERROR',
WORD,   HOLDS CURRENT WORD OF INTERPRETIVE CODE. REFERENCING THIS
        INDIRECTLY GETS A VALUE WHICH IDENTIFIES THIS LANGUAGE ELEMENT,
            0000            MEANS   A LINE NUMBER
            1000            MEANS   A VARIABLE
            2000-2003       MEANS   A LITERAL
            3000            MEANS   QUOTED TEXT
            4000-5777       MEANS   A SYSTEM SYMBOL SUCH AS '+' OR 'STEP'
            6000-7777       MEANS   A STATEMENT WORD SUCH AS 'LET' OR 'GOTO'
024..,  LITERAL,
LINENO, POINTER TO THE LAST LINE NUMBER ELEMENT ENCOUNTERED AT THE
        BEGINNING OF A LINE,
(OTEMP, TEMPORARY USED IN GOTO AND GOSUB,
(774?,  LITERALS,
(777?,
054..,
EPTR,   DURING THE LINE TRANSLATION PROCESS, THIS POINTS TO THE FIRST
        CHARACTER WHICH IS TO BE TRANSLATED,
FPTR,   DURING THE LINE TRANSLATION PROCESS, THIS POINTS TO
FPTR,   DURING THE LINE TRANSLATION PROCESS, THIS POINTS TO WHERE THE
        TRANSLATED SYMBOL GOES,
```

WITH AN ASSOCIATED ENTRY IN THE VARIABLE SPACE:

```
        +-------
        !       THE VARIABLE NAME(TWO CHARACTERS OF TRIMMED ASCII.)
        +-------
        !       HIGH WORD OF THE SCALAR CELL
        +-------
        !       MIDDLE WORD OF THE SCALAR CELL
        +-------
        !       LOW WORD OF THE SCALAR CELL
-----   +-------
  A     :       HIGH WORD OF ARRAY(0) OR ARRAY(0,0)
  !     +-------
  !     !       MIDDLE WORD OF ARRAY(0) OR ARRAY(0,0)
  !     +-------
  !     :       LOW WORD OF ARRAY(0) OR ARRAY(0,0)
OPTIONAL+-------
  !     !       HIGH WORD OF ARRAY(1) OR ARRAY(0,1)
  !     +-------
  V     !       ETC.
-----   +-------
```

TEXT STRINGS(AS IN PRINT STATEMENTS) ARE TRANSLATED INTO A SPECIAL POINTER FOLLOWED BY TRIMMED ASCII ENDING WITH A ZERO CHARACTER. "NOW IS THE TIME" TRANSLATES INTO:

XXTEXT!4216!1727!4011!2340!2410!0540!!411!1505!4200

IF END OF LINE IS ENCOUNTERED BEFORE A SECOND QUOTE, END OF TEXT IS FORCED.

LITERALS(FLOATING POINT NUMBERS IN BASIC) ARE TRANSLATED INTO A SPECIAL POINTER FOLLOWED BY FROM ZERO TO THREE WORDS OF THE ACTUAL OCTAL OF THE LITERAL.

0	TRANSLATES INTO	XXLIT0
1.5	TRANSLATES INTO	XXLIT1!2016
5.5	TRANSLATES INTO	XXLIT2!2035!4000
65535	TRANSLATES INTO	XXLIT3!2207!7777!4000

THE NUMBER OF WORDS COMPRISING THE LITERAL IS DETERMINED BY WHETHER XXLIT0, XXLIT1, XXLIT2 OR XXLIT3 PRECEDES IT. THE LITERAL IS FILLED OUT WITH ZERO BITS.

PROGRAM NARRATIVE

AVRLC, POINTER TO THE NEXT FREE WORD IN VARIABLE SPACE.
VARIABLES RANGE UP FROM THE TOP OF THE SYSTEM(ABOUT 5400).
CODELOC,POINTER TO THE BEGINNING OF THE INTERPRETIVE CODE.
PSYMTAB,POINTER TO THE BEGINNING OF THE USER SYMBOL TABLE.
NSYMTAB,ONES COMPLEMENT OF THE NUMBER OF ENTRIES IN THE USER SYMBOL TABLE.

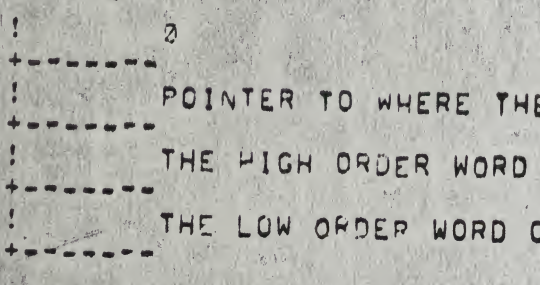
```
*(USER VAR.)                /USERR VARIABLE SPACE
VARX,   TEXT /X/            /       VARIABLE NAMED 'X'
        Ø                   /       VALUE  Ø,Ø
        Ø
        Ø
VART,   TEXT /T/            /       VARIABLE NAMED 'T'
        Ø                   /       VALUE  Ø,Ø
        Ø
        Ø
VARI,   TEXT /I/            /       VARIABLE NAMED 'I'
        Ø                   /       VALUE  Ø,Ø
        Ø
        Ø
```

THE DESIGN OBJECTIVE WAS THAT THE INTERPRETIVE CODE SHOULD BE ONE WORD PER SYNTACTIC ELEMENT.(ONE WORD PER VARIABLE REFERENCE, ONE WORD PER LITERAL, ONE WORD PER KEYWORD, ONE WORD PER SPECIAL SYMBOL.) IN THE WORKING PROGRAM,THIS IS ACTUALLY THE CASE EXCEPT FOR TWO LANGUAGE ELEMENTS - LITERALS, AND TEXT STRINGS,

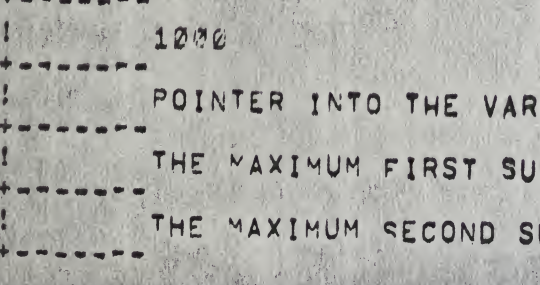
SYMBOLS LIKE + - * <= INT INPUT ETC, ARE TRANSLATED INTO POINTERS INTO THE PERMANENT SYMBOL TABLE. THE PERMANENT SYMBOL TABLE IS LOCATED ABOUT 3 PAGES FROM THE TOP OF CORE, AND THE SYMBOLS HAVE NAMES SUCH AS XXPLUS,XXMINUS,XXSTAR,XXLE,XXINT,XXINPUT ETC.
(END OF LINE TRANSLATES INTO THE SYMBOL 'XXCRLF'.)
THESE ACCOUNT FOR ABOUT HALF THE WORDS IN THE CODE SECTION OF THE SAMPLE PROGRAM.

LINE NUMBERS AND VARIABLES ARE TRANSLATED INTO POINTERS INTO THE USER SYMBOL TABLE, WITH ENTRIES ADDED WHEN NECESSARY. ALL ENTRIES IN THE USER SYMBOL TABLE ARE 4 WORDS LONG.

THE CONTENTS OF A LINE NUMBER ENTRY ARE:



THE CONTENTS OF A VARIABLE NAME ENTRY ARE:



```
        XXEOF           /       (EOF)


*(USER TABLE)           /LINE NUMBER AND VARIABLE TABLE
SYM100, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF100          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        144
SYMX,   1000            /       THIS ENTRY IS A VARIABLE
        VARX            /       ITS DEFINITION
        0               /       IT IS A SCALAR
        0
SYM104, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF104          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        150
SYMT,   1003            /       THIS ENTRY IS A VARIABLE
        VART            /       ITS DEFINITION
        0               /       IT IS A SCALAR
        0
SYM110, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF110          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        154
SYM140, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF140          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        214
SYM120, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF122          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        174
SYM130, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF130          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        202
SYM160, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF160          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        240
SYMI,   1000            /       THIS ENTRY IS A VARIABLE
        VARI            /       ITS DEFINITION
        0               /       IT IS A SCALAR
        0
SYM170, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF170          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        252
SYM180, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF180          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        264
SYM190, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF190          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
        274
SYM200, 0               /       THIS ENTRY IS A LINE NUMBER
        DEF200          /       ITS DEFINITION
        0               /       ITS PRINT VALUE(24 BITS)
```

```
        XXCRLF          /       (EOL)
DEF140, SYM140          /       140
        XXIF            /       IF
        SY1T            /       T
        XXLE            /       <=
        XXLIT1          /       1.3
        2314
        XXTHEN          /       THEN
        SYM150          /       160
        XXCRLF          /       (EOL)
DEF150, SYM130          /       150
        XXLET           /       LET
        SY1T            /       T
        XXEQ            /       =
        XXLIT1          /       1.0
        2314
        XXSLASH         /       /
        SY1T            /       T
        XXCRLF          /       (EOL)
DEF160, SYM150          /       160
        XXFOR           /       FOR
        SY1I            /       I
        XXEQ            /       =
        XXLIT1          /       1.0
        2314
        XXTO            /       TO
        XXLIT1          /       5.0
        2315
        XXSTEP          /       STEP
        XXLIT1          /       2.0
        2324
        XXCRLF          /       (EOL)
DEF170, SYM170          /       170
        XXLET           /       LET
        SY1X            /       X
        XXEQ            /       =
        SY1X            /       X
        XXPLUS          /       +
        SY1T            /       T
        XXSLASH         /       /
        XXOPEN          /       (
        SY1T            /       T
        XXPLUS          /       +
        SY1I            /       I
        XXCLOSE         /       )
        XXCRLF          /       (EOL)
DEF180, SYM180          /       180
        XXNEXT          /       NEXT
        SY1I            /       I
        XXCRLF          /       (EOL)
DEF190, SYM190          /       190
        XXPRINT         /       PRINT
        XXTEXT          /       "OUTPUT IS"
        TEXT /"OUTPUT IS"/
        XXSEMI          /       ;
        SY1X            /       X
        XXCRLF          /       (EOL)
DEF200, SYM200          /       200
        XXEND           /       END
        XXCRLF          /       (EOL)
```

EDUSYSTEM 10 (4K BASIC)

THE SYSTEM TRANSLATES EACH LINE AS IT IS TYPED INTO AN INTERNAL FORMAT SUITABLE FOR FAST INTERPRETATION AT RUN TIME. SINCE THIS FORMAT IS BASIC TO THE UNDERSTANDING OF THE PROGRAMMING, IT WILL BE DESCRIBED FIRST. TO ILLUSTRATE THE TRANSLATED FORM, HERE IS A BASIC PROGRAM AND ITS INTERNAL REPRESENTATION:

```
100 INPUT X
104 LET T=ABS(X)
110 IF T>2**-15 THEN 140
120 T=X
130 STOP
140 IF T<=1 THEN 160
150 LET T=1/T
160 FOR I=1 TO 5 STEP 2
170 LET X=X+T/(T+I)
180 NEXT I
190 PRINT "OUTPUT IS";X
200 END
```

```
*(CODE)
                        /INTERPRETIVE CODE
DEF100, SYM100          /       100
        XXINPUT         /       INPUT
        SYMX            /       X
        XXCRLF          /       (EOL)
DEF104, SYM104          /       104
        XXLET           /       LET
        SYMT            /       T
        XXEQ            /       =
        XXABS           /       ABS
        XXOPEN          /       (
        SYMX            /       X
        XXCLOSE         /       )
        XXCRLF          /       (EOL)
DEF110, SYM110          /       110
        XXIF            /       IF
        SYMT            /       T
        XXGT            /       >
        XXLIT1          /       2.0
        2**-14
        XXUPARR         /       ↑
        XXMINUS         /       -
        XXLIT2          /       15.0
        2047
        4000
        XXTHEN          /       THEN
        SYM140          /       140
        XXCRLF          /       (EOL)
DEF120, SYM120          /       120
        SYMT            /       T
        XXEQ            /       =
        SYMX            /       X
        XXCRLF          /       (EOL)
DEF130, SYM130          /       130
        XXSTOP          /       STOP
```

When loading Basic/RT from field 1, it will automatically start up. When a user function or command is to be used, two binary tapes must be loaded in which case the automatic startup is undesirable. If the LAB8/e has 12K or more, put the binary loader in field 2. Load both tapes and start Basic/RT in field Ø, data field Ø at location 1000. In an 8K machine, load Basic/RT and let it start up. Hit a control C, halt the machine and load the overlay tape via the binary loader. Restart Basic/RT at 1000 or 6725 of field Ø, data field Ø. Note, when loading the overlay tape with the binary loader, make sure the data field is set correctly.

In conclusion, to do functions or user commands more complicated then shown above, the correct subroutine calling sequence can be established from the listing and the attached documentation. Basic/RT was created from 4K Basic Edusystem 1Ø. There is documentation for 4K Basic Edusystem 1Ø. The Basic/RT documentation considers only the changes from 4K Basic to 8K Basic/RT.

jkw

EXAMPLE 4. User command in field 1. In this example, UCOM will type a Ø on the teletype (it may overtype the previous line).

```
/PAS04                                  PAL8-V7          PAGE 1

                    /PAS04
            0000                FIELD 0
            0306                *306
00306       7743                7743
            0001                FIELD 1
            0054    UJMP=54
            7175    DEVCOM=7175
            0744                *744
10744       0065                UCOM
            0065                *65
10065       7200    UCOM,       CLA
10066       6002                IOF
10067       6041                TSF
10070       5067                JMP .-1
10071       1077                TAD UC260
10072       6046                TLS
10073       6001                ION
10074       7200                CLA
10075       4454                JMS I UJMP
10076       7175                DEVCOM
10077       0260    UC260,      260

$
```

```
                    /BAS03
            0016    AC3=16
            4744    FIX=4744
            0177    GETWD=177
            7616    MEVAL=7616
            4435    FENTER=4435
            0200    FWD=200
            2000    FSTA=2000
            0000    FEXIT=0
            6000    FMU=6000
            7176    DEVCON=7176
            0306            *306
00306       6114            UCOM
            6114            *6114
06114       4740    UCOM,   JMS I XGETWD
06115       4741            JMS I XMEVAL
06116       4435            FENTER
06117       2226            FSTA+FWD+TEMP-.
06120       0000            FEXIT
06121       4741            JMS I XMEVAL
06122       4435            FENTER
06123       6222            FMU+FWD+TEMP-.
06124       0000            FEXIT
06125       4742            JMS I XFIX
06126       7200            CLA
06127       1016            TAD AC3
06130       1343            TAD UC260
06131       6002            IOF
06132       6041            TSF
06133       5332            JMP .-1
06134       6046            TLS
06135       6001            ION
06136       7200            CLA
06137       5744            JMP I XDEVCO        / exit.
06140       0177    XGETWD, GETWD
06141       7616    XMEVAL, MEVAL
06142       4744    XFIX,   FIX
06143       0260    UC260,  260
06144       7176    XDEVCO, DEVCON
06145       0000    TEMP,   0;0;0
06146       0000
06147       0000
```

3. Writing a user command in field Ø: The section in the Basic/RT user manual or the Basic/RT section of the new manual describes how to implement a user command. However, there is no available room in field Ø and some function or command must be eliminated to make room. To implement a user command in field Ø, store UCOM at location 306 of field Ø and write the command over an existing function. Example 3 writes the command over the ExP function.

   Return to Basic by jumping to DEVCON, DEVCOM or SKIPIT as described in the Basic user's manual.'

4. Writing a user command in field 1: To put the command in field 1, put a 7743 at location 306 of field Ø and the UCOM address at 744 of field 1. At the end of the UCOM code exit back to field Ø by doing a

```
JMS I UJMP
DEVCON
```

   or

```
JMS I UJMP
DEVCOM
```

   or

```
JMS I UJMP
SKIPIT
```

   depending on the condition of the user command as explained in the Basic/RT user's manual. When a user's command UCOM is entered the data field is for field 1 (CDF 1Ø). On exit the data field must also be field 1. Remember the floating point package can not be called from field 1.

EXAMPLE 3. User command in field Ø. In this example, UCOM A,B will multiply A times B, fix the answer, add 260 octal to the value and issue a TLS.

```
UUF2,  Ø
UUF3,  Ø
UUF4,  Ø
UUF5,  Ø
UM12, -12
UFLDØ, 6114
UFLD1, UUFX

UUFX,  Ø
       4435        /FENTER
       2205        /FST+FWD+X-.
       6204        /FMP+FWD+X-.
       0000        /FEXIT
       CDF CIF 1Ø
       5714        /JMP I UUF
       Ø           /X
       Ø
       Ø
```

In this example the user function was moved to field Ø, so that it could use the floating point package. The only restriction is that the UUF can't be called at the same time the LOG function is called. The example isn't a very good programming technique, but is meant only to demonstrate the calling sequence of a UUF in field 1.

EXAMPLE 2. UUF(A)=A^2

UUF function in field 1

```
        FIELD Ø
        *1156
        7747'

        FIELD 1
        *764
        UUF

        *65

UUF,    Ø

        JMS UUF1        /interchange field Ø + 1 code
        CDF CIF Ø
        JMS I UFLDO     /call UUF in field Ø
        JMS UUF1        /reexchange code
        CDF 1Ø
        JMP I UUF       /exit from UUF

UUF1,   Ø

        TAD UM12        /load number of words to exchange
        DCA UUF2
        TAD UFLD1       /set field 1 and Ø addresses
        DCA UUF3
        TAD UFLDØ
        DCA UUF4

UUF6,   CDF 1Ø          /exchange values
        TAD I UUF3
        DCA UUF5
        CDF Ø,
        TAD I UUF4
        CDF 1Ø
        DCA I UUF3
        CDF Ø
        TAD UUF5
        DCA I UUF4

        ISZ UUF3        /get next word
        ISZ UUF4
        ISZ UUF2
        JMP UUF6

        JMP I UUF1
```

EXAMPLE 1. UUF(A)=A^2

UUF function in field ø

```
        FENTER=4435
        FST=2000
        FWD=200
        FMP=6000
        FEXIT=0000
        FIELD ø
        *1156
        UUF

        *6114

UUF,    ø
        FENTER                  /ENTER FPP
        FST+FWD+X-.             /Save N
        FMP+FWD+X-.             /N*N
        FEXIT
        JMP I UUF

X,      ø
        ø
        ø

        $
```

COULD "LOCK OUT" BASIC SO NOTHING COULD HAPPEN, THAT IS
WHY WE MUST CHECK FOR ^C HERE, IF THIS IS THE CASE
THEN WE MUST TAKE THE SAME ACTION AS FOR THE CASE WHERE
WE COULD PHYSICALLY DETECT THE FACT THAT THE CLOCK WAS
RUNNING TOO FAST. WHEN WE DECIDE THAT THE CLOCK IS RUNNING TOO FAST,
WE STOP THE CLOCK (TO PREVENT FURTHER CLOCK INTERUPTS) AND
CALL 'RTERR' TO PRINT OUT THE ERROR MESSAGE ["RATE ERROR"];
IF THERE WAS NO ERROR, THEN WE EXIT NORMALLY VIA 'INTEXT'.

APUT: THIS PUTS A SAMPLE IN THE BUFFER, IF NO ROOM, WE GET
A BUFFER FULL MESSAGE, POINTERS ARE SET UP BY REAL TIME STATEMENT.

AGET: OPPOSITE OF APUT, IF NOTHING THERE, IT PUTS BASIC TO
SLEEP UNTIL AN INTERUPT OCCURS', DESCRIBED PREVIOUSLY,

ABOP: THIS BOPS UP POINTERS OF APUT AND AGET, LIMITS SET BY
REAL TIME STATEMENT, ADA1,ADA2 AND ADA3 ARE DETERMINING
LIMITS AND FACTORS,

UCLS: CLS FUNCTION, PICKS UP CLKSTS AND RETURNS IT,

UCLC: EXECUTES 6137 AND RETURNS RESULTS IN FAC,

UUULLL: THIS IS THE UPPER CORE RESETER',

UUAD: TAKES AC AS CHANNEL NUMBER AND DOES CONVERSION, HAS
TIME OUT INCASE NO A-D OR A-D NOT WORKING,

USETF: THIS ROUTINE TAKES THE AC AND MAKES IT INTO A GOOD FAC,
IT CALLS BEGFIX AND ANORM,

UUAC1-3: THESE ROUTINES ADD AC1 -AC3
RESPECTIVELY TO THE AC, THEY GET IT FROM FIELD Ø,

UUMEVAL: MEVAL FUDGE CALL

UUFIX: FIX FUDGE CALL,

UUDEVC: DEVCON FUDGE CALL,

UUJMS: UPPER CORE COUNTER PART OF LLLJMS

UUJMP: UPPER CORE COUNTER PART OF LLLJMP

UPCOMDO: UPPER CORE COMMAND DISPATCHER, VERY SIMPLE,

UPFUN: UPPER CORE FUNCTION DISPATCHER', ALSO SIMPLE,

CALLS DBPUT TO PUT IN WORD. IT THEN PUTS IN A 4000 TO END THE BUFFER.

DBPUT: THIS ROUTINE PUTS THE AC IN THE BUFFER. IF THE CONTENTS OF WHERE IT'S GOING=4001, THEN WE'VE RUN OUT OF ROOM AND A "TOO-BIG" MESSAGE IS GIVEN. IT RETURNS TO CALLER IF SUCCESSFULL.

UUDATA: THIS GIVES THE "A-D FULL" MESSAGE.

SETUP: THIS IS ORIGINALLY EDUSYS-10 CODE WHICH WAS MOVED BECAUSE WE NEEDED ROOM ELSEWHERE.

ISET: THIS ROUTINE INITIALIZES THE DISPLAY.

MEVAL: THIS ROUTINE EVALUATES AN EXPRESSION.
LEADING COMMAS ARE IGNORED, BAD SYNTAX GETS AN ERROR, RETURN IS TO CALLER WITH RESULTS IN FAC.

LLLJMS: THIS ROUTINE PROVIDES COMMUNICATION BETWEEN THE UPPER AND LOWER BANKS. TYPICAL CALL FROM UPPER IS

```
JMS     LLLJMS
YYYY
```

WHERE YYYY IS THE ROUTINE THE USER WISHES TO CALL IN LOWER BANK RETURN IS ALWAYS TO UPPER FIELD.

LLLJMP: SAME AS LLLJMS EXCEPT THAT LLLJMS JMS'S AND LLLJMP JMP'S.

UNOAD: GIVES THE "NO A-D" MESSAGE.

CCINTK: THIS ROUTINE CHECKS THE CLOCK FLAG. IF UP, THEN IT GOES TO CLOCK1 IN FIELD 1.

TST: DESCRIBED PREVIOUSLY.

- - - - -

FIELD 1

CLOCK1: THIS ROUTINE IS EXECUTED ON A CLOCK INTERUPT.
IT FIRST CLEARS THE FLAG AND SAVES THE STATUS IN CLKSTS.
IT THEN INCREMENTS THE TIME OF DAY [TIM2 AND TIM1],
THEN CHECKS ADACPT AND ABDGET TO SEE IF WE'RE SAMPLING.
IF WE ARE, THEN IT PICKS UP THE CHANNEL [ADCX] AND CALLS
DOAD TO SAMPLE AS MANY TIMES AS NECESSARY AS DETERMINED BY
ADCONT. IT CALLS APUT TO PLACE THE SAMPLES IN THE
BUFFER. IT THEN INCREMENTS CT3,CT2 AND/OR CT1 TO FIND OUT
WHETHER OR NOT DONE. IF DONE, IT CLEARS ADACPT AND SETS ABDGET
MINUS.
IT NOW CHECKS TO SEE IF THE CLOCK IS UP AGAIN. IF IT
IS , WE GET A RATE ERROR. IF IT ISN'T, WE CHECK FOR ^C TO SEE
IF IN THE REGION WHERE WE ARE INTERUPTING SO FAST THAT
WE CANNOT CHECK FOR A ^C IN THE TTY ROUTINES, BUT NOT FAST ENOUGH
SO THAT THE CLOCK FLAG TEST WILL CATCH IT. THIS CONDITION

DONE. IF INPUT IS FROM THE TELETYPE, ALL IS AS BEFORE.

FPT: THE FLOATING POINT PACKAGE IS AS BEFORE, EXCEPT FOR SOME COE'S. THE ONLY TRICKY AREA IS THAT INDIRECTS GO THROUGH FIELD ONE, SO USE CAUTION.

TAB: DESCRIBED PREVIOUSLY (AS WITH TABDO).

INTER: THIS IS THE MAIN INTERUPT PROCESSOR. IT IS DESCRIBED ABOVE. THE ONLY THING NOT MENTIONED WAS BIDLE.

BIDLE: BIDLE IS A ROUTINE TO BE CALLED WHEN YOU WANT TO PUT BASIC TO SLEEP. ESSENTIALLY THIS SETS A FLAG CALLED BUSY. WHEN BASIC GOES TO EXIT FROM AN INTERUPT,
IT CHECKS THE STATUS OF BUSY. IF IT'S ZERO, IT EXITS NORMALLY. THIS MEANS IT WAS IN BASIC WHEN IT INTERUPTED. IF IT'S ONE, THEN IT WAS IN THE NULL JOB ROUTINE. IF SO, THEN NULLJOB COULD ONLY BE STARTED BY BIDLE. HENCE AFTER SERVICING THE INTERUPT, IT RETURNS FROM BIDLE AND CLEARS THE BUSY FLAG. WHEN BIDLE IS CALLED AGAIN, BIDLE WILL EXIT TO THE LOCATION AS DEFINED BY WHAT THE INTERUPT ROUTINES SAVED WHEN IN THE LAST BUSY 1 STATE. HENCE BUSY IS THE WORD WHICH KEEPS TRACK OF WHAT IS GOING ON.

INTEXT: THIS ROUTINE EXITS FROM AN INTERUPT. IT INTEROGATES 'BUSY' TO FIND OUT WHAT TO DO.

PUTER: DESCRIBED PREVIOUSLY.

OBOP: BOPS UP EITHER INPUT OR OUTPUT BUFFER FLAG.

RESET1: THIS RESETS ALL DEVICES. SEE PRESET.

RESET2: THIS IS A MASTER RESET OF ALL HARDWARE STATUS FLAGS. ONLY CALLED WHEN STARTING UP BASIC.

OUTCHL: DESCRIBED ABOVE.

DEVCOM: GETS NEXT ITEM FROM USER STATEMENT, THEN CALLS DEVCON.

DEVCON: CHECKS TO SEE IF ITEM IS A C,R, IF NOT, THEN A SYNTAX ERROR RESULTS.

RTERR: THIS GIVES THE "RATE ERROR" MESSAGE.

GETARY: THIS ROUTINE LOOKS AT USER'S VARIABLE ARGUMENT, IT THEN PICKS UP SUBSCRIPTING INFORMATION, IT THEN CALLS GETADD TO RETURN LAST ELEMENT IN ARRAY. IT HAS THE FIRST, AND NOW THE LAST, SO IT HAS THE SIZE, IT LEAVES POINTER TO FIRST IN AC1 AND POINTER TO LAST IN AC2. THEN IT RETURNS.

PLOTB: THIS SETS UP DISPLAY BUFFER. IT SETS UP DISB TO POINT TO CORRECT LOCATION. IT PUTS A 4001 IN LAST WORD OF BUFFER, IT PUTS A 4000 IN FIRST WORD OF BUFFER SO NOTHING WILL BE DISPLAYED. THEN IT RETURNS.

DBLIT: THIS ROUTINE TAKES THE FAC AND STICKS IT IN THE BUFFER. IT FIRST CHECKS TO SEE IF IN RANGE (0 TO .9999999999) AND THEN IF NOT IT SETS DBAD. IT THEN FIXES FAC, AND THEN

- - - -

IMPORTANT ROUTINES AND LOCATIONS.

I) PAGE ZERO LOCATIONS.

0-2 USED FOR INTERUPT VECTOR.

DISAUTO, AUTO REGISTER USED AS POINTER TO NEXT ITEM TO BE DISPLAYED (EITHER AN X OR Y POINT). FOUND IN NULJOB.

INDEV, CONTAINS INPUT DEVICE NUMBER (1=TTY, 2=PTR)

OUTD?, CONTAINS OUTPUT DEVICE WHILE ACCEPTING AND ECHOING INPUT FROM THE TTY. IN OTHER WORDS, A TEMPORARY ASSIGNED JUST TO HOLD THE 'REAL' OUTPUT DEVICE.

OUTDEV, OUTPUT DEVICE THE ROUTINES WILL USE (0=NOTHING, 1=TTY, 2=PTP, 3=LPT)

ODEV, OUTPUT DEVICE CURRENTLY BEING USED BY THE BUFFER.

CNTLC, THE CONTROL O FLAG (+O). IF 0, NO CONTROL O.

NOINT, A LOCATION WHICH TELLS WHETHER OR NOT THE ^C CHARACTER CAN INTERUPT BASIC. CERTAIN PROCESSES (SUCH AS CORE JUGGLING) CANNOT BE INTERUPTTED. THEREFORE THEY SET THE NOINT SWITCH TO A 4001. WHEN DONE, THEY RESET IT TO A 0. IF A ^C SHOULD BE STRUCK, IT WILL NOT STOP BASIC, BUT IT WILL SET THE NOINT SWITCH TO A 7777. WHEN WE LEAVE THE CRITICAL PROCESS, NOINT WILL BE CHECKED TO SEE IF IT CONTAINS A 7777, AND IF SO, WE PROCEED AS IF A CONTROL C HAD JUST BEEN STRUCK.

RBSWCH, THIS IS THE RUBOUTS SWITCH. 0=IGNORE RUBOUTS, <> TREAT RUBOUTS LIKE BACK-ARROWS (←), WHICH IS SHIFT O.

DISB, ADDRESS OF PRESENT DISPLAY BUFFER. IF NONE IS PRESENT, THIS LOCATION IS ZERO. BUFFER IS ALWAYS IN UPPER CORE.

PRESET, THIS POINTS TO A ROUTINE WHICH RESETS ALL FLAGS BACK TO TTY INPUT AND OUTPUT AND CLEANS UP THE BUFFERS. NECESSARY AFTER AN ERROR OCCURS.

SPECINT, THIS POINTS TO SUBROUTINE WHICH CHECKS FOR ADDITIONAL DEVICE INTERUPTS BESIDES THE STANDARD ONES. THE CLOCK TEST IS LOCATED THERE.

PCOMT, THIS POINTER POINTS TO A ROUTINE WHICH WAITS FOR OUTPUT TO TERMINATE (THE BUFFER EMPTY). THIS IS NECESSARY BECAUSE OTHERWISE ERROR AND/OR LISTING CONTROL INFORMATION MIGHT BE LOST IF A CONTROL O IS HIT AT THE WRONG TIME. IT ALSO RESETS THE CONTROL O FLAG. THIS ROUTINE IS ALSO SPEAKILY USED BY THE LIST * OPTION TO SET THE NULL AFTER CARRIAGE RETURN FLAG [PUTXRA]. IT STORES THE AC IN PUTXRA FIRST, THEN WAITS FOR OUTPUT TO TERMINATE. HENCE THIS IS A TWO FOLD ROUTINE.

DELOUT, THIS POINTS TO A ROUTINE WHICH DELETES THE CURRENT CONTENTS OF THE OUTPUT BUFFER, USED FREQUENTLY BY MANY ROUTINES.

CNCLR, THIS ROUTINE TESTS AND CLEARS THE NOINT FLAG DISCUSSED EARILER.

SPLEET, THIS POINTER POINTS TO SUBROUTINE WHICH DETERMINES WHETHER OR NOT A GIVEN OPERATION WILL OVERFLOW FREE CORE. THE AMOUNT OF ROOM YOU WISH TO TAKE SHOULD BE IN THE AC. IF NO ROOM, THEN AN ERROR MESSAGE IS GIVEN.

M) THE WAIT COMMAND.
THIS IS INTERNALLY IDENTICAL TO THE WAITC COMMAND. IT "FUDGES" THE TIME [TIM2] SO THAT WHEN AN INTERUPT OCCURS, IT LOOKS LIKE THE TIME HAS RUN OUT AND RETURNS.

N) THE ACCEPT COMMAND.
THE ACCEPT COMMAND WILL SET THE ACCEPT DATA SWITCH [ADACPT] IF AND ONLY IF WE HAVE A GOOD, CLEAN BUFFER [ABDGET>0]. IF WE DO NOT, WE FALL THROUGH TO THE 'REJECT' COMMAND.

O) THE REJECT COMMAND.
THE REJECT COMMAND CLEARS THE ACCEPT DATA FLAG [ADACPT]. IT RETURNS TO BASIC VIA DEVCOM.

P) THE REAL TIME COMMAND.
THIS COMMAND IS ONE OF THE MORE HAIRY OF THE NEW LAB-8/E BASIC COMMANDS. THE FOLLOWING IS WHAT OCCURS. IT FIRST CHECKS TO SEE IF WE ARE ACCEPTING DATA. IF WE ARE, THE COMMAND IS IGNORED [VIA SKIPIT]. IF WE ARE NOT ACCEPTING DATA, WE CALL GETARY TO GET US THE AREA THE USER REQUESTED. THIS DONE, WE THEN USE THIS INFORMATION TO SET THE BUFFER POINTERS [APUT1 AND APUT2] AND THE BUFFER LIMIT CONTROL INFORMATION [ADA1, ADA2, AND ADA3]. WE THEN GET THE CHANNEL NUMBER TO START [VIA UMEVAL] AND PLACE IT IN ADCX. WE THEN GET THE NUMBER OF CHANNELS TO USE AND PLACE IT IN ADCUNT. WE FINALLY GET THE NUMBER OF TIME COUNTS TO DO AND LEAVE IT IN CT1, CT2 AND CT3. REMEMBER THAT THE TIME IS DOUBLE WORD, HENCE THE DOUBLE WORD ARITHMETIC AT THIS POINT. CT1 IS USED TO GIVE US A THREE WORD ZERO COUNTER SHOULD THE USER PUT IN 0 AS NUMBER OF COUNTS. NOTE THAT 2.36 IS A VERY LONG TIME. WE THEN ZERO OUT THE NUMBER OF SAMPLES PRESENTLY IN THE BUFFER [ACOUNT] AND TELL THE SYSTEM THAT WE'VE GOT A GOOD BUFFER [ABDGET >0].

Q) THE TIME FUNCTION [TIM]
THE TIME FUNCTION RETURNS THE NUMBER OF TICS, EVERY TIME THE CLOCK INTERUPTS, CLOCKI (THE CLOCK INTERUPT ROUTINE) INCREMENTS TIM2 AND THEN TIM1. TIM RETURNS THIS TIME IN THE AC.

R) THE ADB FUNCTION.
THIS FUNCTION CALLS UADCB WITH THE AC ALL 7777'S. UADCB WILL BE DESCRIBED NOW. UADCB IS THE MASTER A-D FUNCTION SWITCHER. UADCB EITHER DOES AN IMMEDIATE ADC IF THE AC IS ZERO OR WILL ATTEMPT TO GET A SAMPLE FROM THE USER'S A-D BUFFER IF THE AC IS NON-ZERO. IF THE AC IS ZERO, IT WILL CALL DOAD WITH THE INTEGERIZED AC3 (BITS 9-11). DOAD DOES AN A-D CONVERSION ON THE CHANNEL IN THE AC. IT THEN RETURNS THIS VALUE IN THE AC. UADCB THEN GOES TO UINAC, WHERE THE A-D VALUE IS PLACED IN THE FAC, NORMALIZED AND THEN BROUGHT INTO THE CORRECT NUMERIC RANGE (-1.V TO +1.V), THE FUNCTION THEN RETURNS. IF THE AC IS NOT ZERO (FOR AN ADB), IT CHECKS TO SEE IF THE BUFFER IS ACTIVE. IF NOT, AN ERROR IS GIVEN. IT THEN CHECKS TO SEE IF THE BUFFER IS ACTIVE, BUT THE TIME HAS RUN OUT. IF SO, ANOTHER MESSAGE IS GIVEN. IF ALL IS WELL. IT CALLS AGET TO GET AN A-D VALUE FROM THE USERS BUFFER. AGET IS MERELY THE RING BUFFER REMOVER. IF NOTHING IS IN THE BUFFER [ACOUNT=0], THEN AGET WILL PUT BASIC TO SLEEP. AGET RETURNS WITH CONVERTED NUMBER IN AC. UADCB THEN GOES TO UINAC WHICH WAS DESCRIBED ABOVE.

HEAD TO THE DESIRED POSITION. IT CONSIST OF TWO PARTS.
THE FIRST IS THE ACTUAL TAB FUNCTION. THE SECOND IS CALLED
'TABDO' AND IS CALLED BY PRINT AFTER EVALUATING AN EXPRESSION
IF THE TABFLG IS SET. WHAT HAPPENS IS AS FOLLOWS:
WHEN TAB IS CALLED, IT FIRST FIXES THE ARGUMENT, IT THEN SETS
'TABFLG' TO INDICATE THAT WE HAVE PROCESSED A TAB FUNCTION.
IT THEN GETS THE PRESENT POSITION OF THE PRINT HEAD FROM
'COLUMN' AND RETURNS THIS AS THE FUNCTION VALUE. THUS TAB CAN
BE USED AS A REGUALR FUNCTION. IT THEN RETURNS TO THE CALLER.
THE PRINT STATEMENT PROCESSOR CHECKS TABFLG AFTER EVALUATING AN
EXPRESSION. IF IT'S SET (NON-ZERO) THEN IT GOES TO TABDO. TABDO
PICKS UP WHERE WE WANT TO GO TO (LEFT IN 'TABDES' BY TAB) AND
FIGURES OUT WHERE WE ARE BY USING COLUMN. IF WE ARE TO FAR IT
GIVES A CARRIAGE RETURN AND A NULL CHARACTER(TO PREVENT
TIMING PROBLEMS). IT THEN SPACES OVER THE CORRECT NUMBER OF
SPACES. WHEN DONE IT RETURNS TO 'PRINT+1' TO
FINISH PROCESS THE COMMAND. NOTE THAT WE CANNOT RETURN TO THE
SECTION WHICH CALLED IT BECAUSE IT WILL PRINT OUT A VALUE (
SUCH AS 23) BECAUSE IT WILL TAKE THE EVALUATED NUMBER AND
CONVERT IT TO ASCII.

THE FOLLOWING COMMANDS ARE ALL DONE IN (RESIDE) IN FIELD 1.

THE ADC FUNCTION.
THIS FUNCTION [RENAMED ZZADC] CALLS THE ROUTINE UADCB WITH
A ZERO AC. UADCB IS THE GENERAL A-D
DISPATCHER. UADCB HAS TWO COURSES OF ACTION DEPENDING ON THE AC.
THE USER SHOULD READ IT'S DESCRIPTION AT
THE END. UADCB WILL RETURN WITH THE CORRECT ADC VALUE IN THE
FAC. IT WILL THEN RETURN TO THE CALLER.

THE SET RATE COMMAND.
THIS COMMAND SETS THE CLOCK GOING FROM THE ARGUMENTS. IT
FIRST CALLS MEVAL [VIA UMEVAL] TO GET THE RATE (0-7). IT THEN
PLACES THIS IN THE COMMAND REGISTER ALONG WITH A 5010. THESE ARE
THE CORRECT HARDWARE FLAG BITS TO RUN IN 'NORMAL' CLOCK MODE.
IT THEN JUMPS TO USETM. USETM WILL EVALUATE THE NEXT EXPRESSION
[VIA UMEVAL] AND WILL THEN LOAD THE CLOCK REGISTER
WITH THIS NUMBER. IT THEN CLEARS THE TWO TIME REGISTERS (TIM1
AND TIM2). THESE ARE INCREMENTED EVERY TIME A CLOCK TIC OCCURS.
IT THEN RETURNS TO THE CALLER.

THE SET CLOCK COMMAND.
THIS IS SIMIALIAR TO THE SET RATE COMMAND EXCEPT THAT THE
ENTIRE 12 BITS OF THE MODE THE USER HAS SPECIFIED
IS USED. JUST BIT 8 IS SET ON TO INSURE INTERUPTS. THEN IT GOES
TO USETM.

THE WAITC COMMAND.
THIS COMMAND WAITS FOR A CLOCK TIC. IT FIRST PICKS UP THE LOW
ORDER WORD OF THE CLOCK COUNTER [TIM2] AND SAVES IT IN UTEMP.
IT THEN CALLS BIDLE TO PUT BASIC TO SLEEP. WHEN IT'S AWAKENED.
IT SEES IF THE TIME HAS CHANGED. IF IT HAS, IT RETURNS
VIA DEVCOM. IF IT HASN'T, IT GOES TO SLEEP AGAIN.

A) THE NULL JOB (OR NULJOB). THIS ROUTINE IS CALLED WHENEVER BASIC IS WAITING FOR SOMETHING TO HAPPEN. IT RUNS THE SCOPE IN LAB-8/E BASIC. IT IS ALSO CALLED VIA THE "DELAY" COMMAND. THE DIFFERENCE IS JUST IN THE MANNER OF EXIT. THE FLAG 'NDELAY' CONTROLS THAT. NULJOB CHECKS TO SEE IF THERE IS A PLOTTING BUFFER ASSIGNED. IF THERE IS (DISB IS NON-ZERO) THEN IT WILL SET UP DISAUTO TO BE DISB. IT WILL THEN TAKE OUT X-Y PAIRS OF POINTS AND DISPLAY THEM. A MINUS ENTRY IS THE END OF LIST INDICATOR. ON END OF LIST, IT GOES BACK TO NULJOB AND CHECKS NDELAY TO SEE IF IT SHOULD RETURN TO THE CALLER. IF NOT, IT THEN RESTARTS ALL OVER AGAIN BY RECHECKING DISB.

B) THE PLOT STATEMENT.
WHENEVER A PLOT COMMAND IS ENCOUNTERED, BASIC GOES TO 'PLOT'. PLOT FIRST CHECKS TO SEE IF A BUFFER IS ASSIGNED. IF ONE IS NOT ASSIGNED, IT ASSIGNS SPACE BY CHANGING ARRLOC. IT THEN CALLS PLOTB TO SET UP THE CORRECT BUFFER WORDS. WHEN A BUFFER IS PRESENT, PLOT CALLS MEVAL. MEVAL EVALUATES AN EXPRESSION. IT THEN MULTIPLIES THIS EXPRESSION BY FSHIFT FOR CORRECT SCREEN SCALING. IT THEN CALLS DBLIT TO PLACE IN THE SCOPE BUFFER. IT THEN GOES THE SAME FOR THE Y VALUE (EXCEPT NOT MULTIPLYING). WHEN ALL IS DONE, IT THEN GOES TO DEVCON TO CLEAN UP THE COMMAND.

C) THE USE STATEMENT.
THIS STATEMENT ALLOCATES A SCOPE BUFFER IN BASIC. IT FIRST CHECKS TO SEE IF A BUFFER IS ASSIGNED. IF ONE IS, IT RETURNS. IF ONE ISN'T, THEN IT CALLS GETARY TO FIGURE OUT THE SIZE OF THE ARGUEMENT AND TO ALLOCATE THE SPACE (ACTUALLY, IT'S ALREADY ALOCATED). IT THEN SETS DISB TO
THE CORRECT ADDRESS AND THEN CALLS PLOTB TO SET UP BUFFER. IT THEN EXITS TO DEVCON.

D) THE CLEAR COMMAND.
THIS STATEMENT CAUSES THE SCOPE BUFFER TO BE CLEARED. IF THERE IS A BUFFER PRESENT (DISB<>0), THEN IT PLACES AN ABORT CODE (4000) AS THE FIRST INSTRUCTION IN THE BUFFER. NULJOB WILL STOP DISPLAYING ON THIS, HENCE NOTHING WILL BE DISPLAYED.

E) THE TST FUNCTION.
THE TST FUNCTION TESTS WHETHER OR NOT A CHARACTER HAS BEEN TYPED BY PLACING 'INCHAR' IN AC3 AND THEN NORMAL ZING IT. BECAUSE INCHAR IS ALWAYS ZERO IF NO CHARACTER HAS BEEN TYPED AND IS ALWAYS NON-ZERO IF A CHARACTER HAS BEEN TYPED, THIS MAKES TST BEHAVE AS DESIRED AND DESCRIBED.

F) THE GET AND PUT FUNCTIONS.
THE GET AND PUT FUNCTIONS PERFORM THERE TASKS BY MERELY CALLING PUTCH AND GETCH WITH THE CORRECT ITEM IN THE AC. THEY INSERT-REMOVE THE 8 BIT CHARACTER FROM AC3. FAIRLY TRIVIAL ROUTINES.

G) THE RUBOUTS AND NO RUBOUTS COMMANDS.
THESE COMMANDS SET THE VARIABLE RBSWCH TO A 1 OR 0 DEPENDING ON WHETHER OR NOT TO PROCESS RUBOUTS. THE ROUTINE GETCH TESTS THIS FLAG WHEN INPUTING CHARACTERS.

H) THE TAB FUNCTION.
THIS FUNCTION IS THE FUNCTION WHICH CAN POSITION THE PRINT

THEREFORE, IT IS POSSIBLE TO STORE AWAY CHARACTERS IN IT'S BUFFER TO BE OUTPUT LATER. THE BUFFER IS LOCATED AT OBLOW TO OBHIGH. THEREFORE, TO PREVENT DEVICE MIX-UPS, PUTER HAS IT'S OWN OUTPUT FLAG. IT IS CALLED 'ODEV'. ODEV CONTAINS THE NUMBER OF THE DEVICE WHICH IS NOW USING THE BUFFER. SHOULD PUTER BE CALLED TO PRINT A CHARACTER ON A DEVICE THAT IS DIFFERENT FROM THE ONE WHICH IS NOW USING THE BUFFER, PUTER WILL WAIT (VIA BIDLE) UNTIL THE BUFFER IS EMPTY. THEN IT WILL CONTINUE. THERE ARE TWO OTHER POSSIBLITIES. ONE IS THAT THE BUFFER IS EMPTY, THE OTHER IS THAT THERE ARE CHARACTERS IN IT BUT GOING TO THE SAME DEVICE. IF THE BUFFER IS EMPTY, PUTER 'ASSIGNS' THE BUFFER TO THE DEVICE AND CALLS 'OUTIT' TO OUTPUT THE CHARACTER. OUTIT REMOVES ONE CHARACTER FROM THE BUFFER AND OUTPUTS IT, IF IT'S POSSIBLE. IF THERE ARE CHARACTERS IN THE BUFFER, THEN PUTER MERELY STICKS THE CHARACTER IN THE BUFFER AND RETURNS. THERE IS ONE OTHER FLAG ASSOCIATED WITH THE BUFFER THAT SHOULD CONCERN THE READER, IT IS CALLED 'CNTLO' AND IS THE CONTROL O FLAG. THIS FLAG ESSENTIALLY SAYS WHETHER OR NOT WE ARE UNDER A CONTROL O (SUPPRESS OUTPUT). IF WE ARE INDEED UNDER A CONTROL FLAG, THEN WE DO NOT PROCESS CHARACTERS. INSTEAD WE JUST RETURN IMMEDIATELY.

C) THE INTERUPT CHAIN:

THIS IS THE ROUTINE WHICH CHECKS THE DEVICES ON AN INTERUPT. IT BASICALLY DOES THE FOLLOWING, IT FIRST SAVES THE STATE OF THE MACHINE (AC, LINK, ETC.). IT THEN DOES IOT'S TO FIND OUT WHICH DEVICE IS INTERUPTING. IF IT CANNOT FIND IT, IT HALTS (A FATAL ERROR). IF IT DOES FIND IT, IT GOES TO THE CORRECT ROUTINE. ON AN OUTPUT FLAG (TTY,LPT, PTP) IT CLEARS THE FLAG AND THEN CALLS OUTIT TO PRINT THE NEXT CHARACTER. ON A HIGH SPEED READER INTERUPT, IT PUTS THE CHARACTER IN HRCHAR. ON TTY INTERUPT, IT CHECKS TO SEE IF IT'S A CONTROL CHARACTER (^C OR ^O). IF IT ISN'T, IT MERELY PLACES THE CHARACTER IN INCHAR. IF IT IS A ^O, THEN IT SETS THE CNTLO FLAG AND CALLS 'OUTDEL' TO DELETE THE PRESENT OUTPUT BUFFER. IF IT'S A ^C, IT CHECKS NOINT (DISCUSSED AT THE END). IF IT'S OK TO STOP BASIC NOW, IT THEN RESETS BASIC WITH A STOP-READY MESSAGE AND GOES TO EDIT (THE EDITOR PORTION). THE ONLY OTHER DEVICE IN THE INTERUPT CHAIN IS THE CLOCK, WHICH WILL BE DISCUSSED LATER. TO EXIT FROM AN INTERUPT, 'INTEXT' RESTORES THE MACHINE (AC, LINE, ETC.)

I) LAB-8\E ADDITIONS:

THE LAB-8\E ADDITIONS ARE BASICALLY A SERIES OF COMMANDS WHICH OPERATE ON THE LAB-8\E PERIPHERALS. SINCE THE READER IS FAMILIAR WITH OS/8-10, THE ACTUAL IMPLEMENTAION WILL NOT BE DESCRIBED. JUST THEIR RELATIONSHIP WITH THE REST OF BASIC WILL BE DESCRIBED.

IS PROCESSED, OR WHEN ONE IS TYPED, THE I-O ROUTINES WILL RETURN TO BASIC AND BASIC WILL CONTINUE TO 'RUN' UNTIL IT WANTS ANOTHER CHARACTER OR UNITL IT WANTS TO OUTPUT ANOTHER CHARACTER. IN REALITY, BASIC IS USUALLY WAITING FOR INPUT, WHEN BASIC CALLS THE I-O ROUTINES, THE I-O ROUTINES CHECK TO SEE IF IT CAN DO WHAT BASIC WANTS. IF IT CAN, IT DOES IT AND RETURNS TO BASIC. IF IT CANNOT, THEN IT WILL WAIT FOR AN INTERUPT TO OCCUR AND SEE IF CAN PROCESS IT. IF IT STILL CANNOT, IT WILL WAIT AGAIN UNTIL THE DESIRED CONDTION IS MET. WHILE WAITING, THE I-O ROUTINES 'PUT BASIC TO SLEEP'. THEY DO THIS BY EXECUTING A LITTLE PROGRAM CALLED NULJOB. NULJOB IS RUN WHENEVER THE I-O ROUTINES ARE WAITING FOR AN INTERUPT. IN LAB-8/E BASIC, NULJOB IS THE ROUTINE WHICH DISPLAYS THE CONTENTS OF THE DISPLAY BUFFER.

BASIC CALLS THE I-O ROUTINES TO GET OR PUT A CHARACTER. THE USE OF THE SCOPE WILL BE DISCUSSED IN THE NEXT SECTION.

A) THE GET ROUTINE:

THIS ROUTINE REQUESTS THE I-O ROUTINES TO GET A CHARACTER. THERE IS A FLAG ASSOCIATED WITH THIS ROUTINE. THIS FLAG IS CALLED 'INDEV'. INDEV IS EITHER A 1 OR 2. IF IT'S A ONE, THE GET ROUTINE (CALLED GETCH) WILL GET A CHARACTER FROM THE TTY. IF IT'S A TWO, IT WILL GET IT FROM THE HIGH SPEED READER. THE SEQUENCE OF EVENTS IS AS FOLLOWS:

GETCH FIRST CHECKS THE DEVICE FLAG. IF IT'S A ONE, THEN IT CHECKS TO SEE IF A CHARACTER HAS BEEN TYPED. TYPED CHARACTERS ARE LEFT BY THE INTERUPT ROUTINES IN A LOCATION CALLED 'INCHAR'. IF NO CHARACTER HAS BEEN TYPED, IT CALLS A ROUTINE CALLED 'BIDLE'. BIDLE PUTS BASIC TO SLEEP (RUNS THE NULL JOB) UNTIL AN INTERUPT OCCURS. AFTER BIDLE RETURNS, GETCH AGAIN LOOPS AND CHECKS INCHAR. IF STILL NOT SET, IT PUTS ITSELF TO SLEEP AGAIN.

FOR THE HIGH SPEED READER, THE SEQUENCE IS DIFFERENT. THE ROUTINE 'GWHERE' FIRST REQUESTS A CHARACTER FROM THE HIGH SPEED READER. IT THEN INCREMENTS A COUNTER AND TESTS THE WORD 'HRCHAR'. ON AN INTERUPT FROM THE HIGH SPEED READER, THE INTERUPT ROUTINES PLACE THE CHARACTER IN HRCHAR. IF GWHERE GETS A CHARACTER FROM HRCHAR BEFORE TIME RUNS OUT, IT JUMPS TO THE MIDDLE OF GETCH. IF TIME RUNS OUT, THEN IT PRINTS THE MESSAGE 'TTY' ON THE TTY, AND THEN RESETS INDEV TO 1, AND THEN JUMPS TO NEAR THE BEGINNING OF GETCH TO WAIT FOR A TTY CHARACTER.

ASSUME THAT WE ARE NOW IN GETCH WITH A CHARACTER. WE NOW 'MASK' ALL ALTMODES INTO 175. IF RUBOUTS ARE SELECTED, WE MASK 177'S INTO 137'S. BACKSPACE (210) IS ALSO MASKED INTO 137. THUS SOME PRE-EDITING IS DONE ON THE CHARACTER. IT IS THEN RETURNED IN THE AC FROM GETCH.

) THE PUT ROUTINE:

THIS ROUTINE PRINTS A CHARACTER ON THE DESIRED OUTPUT DEVICE. IT IS ALOT MORE COMPLEX THEN GET. LIKE GET, THERE IS A FLAG ASSOCIATED WITH THE OUTPUT DEVICE. IT IS CALLED 'OUTDEV'. OUTDEV RANGES FROM Ø TO 3. Ø IS NO OUTPUT, 1 IS TTY OUTPUT, 2 IS PTP OUTPUT, AND 3 IS LPT OUTPUT. 'PUTER', THE OUTPUT ROUTINE, IS FULLY BUFFERED.

17734-17777 LEFT ALONE (PRESERVED FOR THE RIM LOADER)

ALL OF THE SPACE THAT WAS IN FREE FIELD 0 IS NOW OCCUPIED BY CODE. THE BINARY LOADER MUST BE IN FIELD ONE DURING LOADING. WHEN BASIC IS STARTED, IT WILL DESTROY IT (BECAUSE IT IS IN THE BUFFER AREA), ONLY THE RIM LOADER WILL BE PRESERVED. LAB-8/E BASIC MAY NOT BE LOADED BY PS/8.

NOW THAT THE CORE LAYOUT HAS BEEN ESTABLISHED, THE MODE OF OPERATION WILL BE DISCUSSED. BASIC IN IT'S NORMAL MODE OF OPERATION WILL BE 'DIDDLING' WITH THE SYMBOL TABLE (EITHER THE PERMANENT ONE OR THE USER'S ONE), OR IT WILL BE EDITING, READING IN, OR TRANSLATING A LINE. THE LINE BUFFER IS IN FIELD ONE, AS IS THE SYMBOL TABLE, FOR-NEXT LIST, GO SUB LIST, ETC. THEREFORE, BASIC WILL BE SPENDING MOST OF IT'S TIME IN FIELD ONE. THIS MEANS THAT BASIC'S NORMAL MODE OF OPERATION IS FOR THE DATA FIELD TO BE SET TO FIELD 1. THEREFORE, THE DATA FIELD IS ALWAYS SET TO FIELD ONE EXCEPT IN THOSE RARE INSTANCES WHERE BASIC MUST REFERENCE FIELD 0 DATA INDIRECT. ONE SUCH PLACE WHERE THIS HAPPENS IS THE FLOATING POINT PACKAGE. THE FLOATING POINT PACKAGE SPENDS MOST OF IT'S TIME IN FIELD ZERO MODE. TO FIND OUT WHERE SUCH SPOTS OCCUR, THE CDF INSTRUCTIONS ARE TAGGED WITH THE COMMENT "/,,,,,,,,,,,, 8K INSERT". THUS IF THE READER IS MODIFYING ONE OF THESE SECTIONS, HE SHOULD BE CAREFUL OF PICKING UP INDIRECT DATA OR EXITING WITH THE DATA FIELD SET INCORRECTLY. ALL TOTALED, THERE ARE RELATIVELY FEW CDF INSTRUCTIONS FOR THE SIZE OF A PROGRAM LIKE BASIC.

II) INTERUPT I/O

WE NOW COME TO THE SECTION WHICH MAKES THE BIGGEST DIFFERENCE BETWEEN EDUSYS-10 AND LAB-8/E BASIC. INTERUPT I/O GIVES LAB-8/E BASIC A 'SIMULATED' BETTER RUN-TIME BECAUSE OF BUFFERING AND ALLOWS THE CLOCK TO BE USED IN A REAL-TIME MANNER. A BASIC OVERVIEW FOLLOWS:

IT WAS STATED IN THE PREVIOUS SECTION THAT BASIC SPENDS MOST OF ITS TIME IN FIELD ONE. THAT IS MOSTLY CORRECT, IF WE TAKE THE WORD 'BASIC' AS MEANING THAT PART WHICH RUNS OR TRANSLATES THE PROGRAM. THERE EXIST OTHER SECTIONS OF THE PROGRAM WHICH RUN WITHOUT EVER REFERENCING ANYTHING IN FIELD ONE. THE I-O ROUTINES ARE ONE OF THESE. THE I-O ROUTINES ARE BASIC'S WAY OF DEALING WITH THE OUTSIDE WORLD. IF BASIC WANTS TO PUT OR GET A CHARACTER, IT CALLS THE I-O ROUTINES TO DO THIS TASK. WHEN THE CHARACTER

LAB-8/E DOCUMENTATION

THIS IS A DESCRIPTION OF THE INTERNAL WORKINGS OF BASIC FOR THE LAB-8/E. SINCE LAB-8/E BASIC IS A MODIFICATION OF EDUSYS-10(OTHERWISE KNOWN AS 4K BASIC), THIS DOCUMENT WILL ONLY DESCRIBE THE CHANGES MADE TO EDUSYS-10 TO PRODUCE LAB-8/E BASIC.

LAB-8/E BASIC WORKS APPROXIMATELY THE SAME WAY AS EDUSYS-10 DOES. IT DIFFERS IN THE FOLLOWING RESPECTS:

1) LAB-8/E BASIC USES 8K OF CORE AS OPPOSED TO 4K OF CORE.
2) LAB-8/E BASIC USES INTERUPTS AND HANDLES MORE DEIVCES.
3) LAB-8/E BASIC HAS ADDITIONAL CODE TO HANDLE THE SPECIAL COMMANDS WHICH PERTAIN TO THE LAB-8/E.

WHAT FOLLOWS IS A LIST OF THE CHANGES MADE TO IMPLEMENT THE ABOVE FEATURES. KNOWLEDGE IS ASSUMED OF BASIC AND EDUSYS-10. A BRIEF DESCRIPTION IS PROVIDED AT THE END OF THIS MEMO WHICH DECRIBES WHAT THE ROUTINES ADDED DO, AND WHAT KEY CORE LOCATIONS MEAN.

1) 8K MODIFICATION.

TO MAKE BASIC USE 8K, THE FOLLOWING CHANGE WAS MADE. IN 4K EVERYTHING RESIDED IN FIELD 0. IN THE 8K VERSION, EVERYTHING FROM THE ARRAY SPACE ON UP NOW RESIDES IN FIELD 1. IN ADDITION, SOME CODE NOW RESIDES IN FIELD ONE TO HANDLE THE CLOCK AND A-D INSTRUCTIONS. THUS A CORE MAP WOULD LOOK LIKE:

0000-10777 BASIC SYSTEM [ALL CODE NEEDED FOR BASIC]
11000-17755 BASIC SYMBOL AREA. THIS AREA IS THE SAME AS IN 4K BASIC: IT CONSISTS OF THE FOLLOWING IN THIS ORDER:

1) ARRAY AND VARIABLE SPACE
2) FREE SPACE (EVERYTHING 'GROWS' INTO THIS SPACE)
3) CODIFIED (COMPILED) BASIC PROGRAM IMAGE.
4) USER SYMBOL TABLE AREA.
5) BASIC'S PERMANENT SYMBOL TABLE AREA.
6) THE LINEBUFFER (TTY AND TRANSLATED LINE BUFFER)
7) THE STACK.
8) THE FOR-NEXT LIST
9) THE GO SUB LIST.

THUS UPPER CORE IS THE AREA WHERE ALL THE DRIVING TABLES AND PROGRAM-VARIABLE AREAS ARE IN BASIC.

```
FATNCJ,
LIMIT,  FIRST LOCATION OF USER FREE SPACE,
        'PLIMIT' IS THE ONLY POINTER TO IT SO THAT CHANGING 'PLIMIT'
        IS ENOUGH TO REMOVE THE FUNCTIONS. THE INITIAL DIALOG
        DOES THIS ON COMMAND.
PERMSYM,THIS IS THE HIGHEST LOCATION OF CHANGEABLE USER STORAGE.
        THE PERMANENT SYMBOL TABLE IS JUST ABOVE IT.

        ITS FORM IS EXACTLY WHAT IT LOOKS LIKE:
        CODE WORD;TEXT 'THE SYMBOLS PRINT NAME'

LINBUF, THE LINE INPUT BUFFER, TRANSLATED LINE STARTS HERE,
        THERE IS NO COUNTER FOR THE FULLNESS OF THE LINE EXCEPT WHEN IT
        IS JUST COLLECTED,
LBEGIN, MORE OF THE LINE INPUT BUFFER, UNTRANSLATED ASCII COMES IN
        STARTING HERE FROM THE TELETYPE,
ENDLIN, THE ASCII CAN EXTEND AS FAR AS 'ENDPDL', BUT THE TRANSLATED
        LINE MUST END AT 'ENDLIN',
PDLIST, PUSH DOWN LIST USED ONLY FOR EXPRESSIONS AND THINGS LIKE THAT,
        'PDL' POINTS INTO THIS,
ENDPDL, END OF THE PUSH DOWN LIST,
FORLIST,FOR TABLE, A MAXIMUM OF 8 TWO WORD ENTRIES,
        A VARIABLE, AND THE LOCATION OF THE 'TO' IN THE 'FOR' STATEMENT,
        'FORCT' IS THE ONES COMPLEMENT OF THE NUMBER OF ENTRIES,
GOLIST, GOSUB TABLE, A MAXIMUM OF 8 ENTRIES,
        A POINTER TO AFTER THE GOSUB,
        'GSBPTR' POINTS INTO HERE TO THE NEXT FREE SPACE,
GSBEND, END OF THE GOSUB TABLE,
        7756 THROUGH 7777 ARE UNTOUCHED BY EXECUTION OF BASIC,
```